MT-TR-9002S

(1)

MAXIM

ACOUSTIC CLASSIFICATION WITH NEURAL NETWORKS

BY

CHARLES S. WEAVER
KRISTINA D. BRANCH
DAVID E. RUMELHART
JOHN S. OSTREM

FINAL REPORT

Contract N00014-89-C286

Submitted To: Office of Naval Research
800 North Quincy Street
Arlington, VA
22217-500

MAXIM TECHNOLOGIES, INC.
3000 Patrick Henry Drive • Santa Clara, California 95054 • (408) 748-1130 • Fax (408) 748-1140

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No 0704-0188

1a REPORT SECURITY CLASSIFICATION: Unclassified	1b RESTRICTIVE MARKINGS: N. A.
2a SECURITY CLASSIFICATION AUTHORITY: N. A.	3 DISTRIBUTION / AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.
2b DECLASSIFICATION / DOWNGRADING SCHEDULE: N. A.	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): MT-TR-9002S	5 MONITORING ORGANIZATION REPORT NUMBER(S): Same
6a NAME OF PERFORMING ORGANIZATION: MAXIM Technologies, Inc. / 6b OFFICE SYMBOL (If applicable):	7a. NAME OF MONITORING ORGANIZATION: Office of Naval Research
6c. ADDRESS (City, State, and ZIP Code): 3000 Patrick Henry Drive, Santa Clara, CA 95-54	7b ADDRESS (City, State, and ZIP Code): 800 N. Quincy Street, Arlington, VA 22217-5000
8a. NAME OF FUNDING / SPONSORING ORGANIZATION: Office of Naval Research / 8b. OFFICE SYMBOL (If applicable): Code 1142PS	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: N00014-89-C-0286
8c. ADDRESS (City, State, and ZIP Code): 800 N. Quincy Street, Arlington, VA 22217-5000	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO: 61153N 42 / PROJECT NO: RR040209 / TASK NO: RR04020901 / WORK UNIT ACCESSION NO: 400o043sbi01

11. TITLE (Include Security Classification)
(U) Acoustic classification with neural networks

12. PERSONAL AUTHOR(S)
Charles S. Weaver, Kristina D. Branch, David E. Rumelhart, John S. Osterm

13a. TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Final	FROM 89/09/15 TO 90/03/14	90/03/14	66

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Acoustic Signal Recognition, Neural Networks, Back Propagation, Adaptive Tree

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

A major objective of the research that is described in this report was the preliminary development and testing of neural networks to aid a sonar operator in detecting and classifying sonar signals and signatures. Two networks were developed to recognize a set of sounds that is representative of sonar pulse type signals. The pattern vector inputs were derived from the signal spectra. Both networks are time delay neural networks that classify by recognizing spectral shapes and the first and second differences of spectral

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT: ☒ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION: Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL: Dr. Harold Hawkins	22b. TELEPHONE (Include Area Code): (202) 696-4502 / 22c OFFICE SYMBOL: Code 1142PS

Block 19 continued:

shapes. One is a back propagation network and the other is a network that does not contain weights and thus is much simpler than back propagation networks to implement in hardware.

Both networks demonstrated a very low false alarm rate and were as accurate or significantly more accurate than the test subjects at signal detection and classification. When the subjects were prompted by a network they were more accurate at finding and classifying the more difficult to detect signals. Our preliminary conclusion is that neural networks will have great value in detecting sonar pulses which is an important signal class.

The architecture and understanding of the weightless network that was developed during Phase I leads us to believe that networks of this type may become important in sonar signal analysis.

TABLE OF CONTENTS

Section

LIST OF FIGURES

LIST OF TABLES

SECTION I
INTRODUCTION AND OVERVIEW

1.0 Introduction

In the Phase I proposal we proposed to study the following question: Can a neural network aid a sonar operator to more reliably and accurately detect and classify sonar signals? An affirmative answer would be important especially when the signals indicate a threat.

To carry out the study, we proposed the following overall technical objectives: (1) To test a Neural Net to determine how the accuracy of a human operator listening to sonar signals is changed when prompted by a neural net that has been trained to recognize sonar signals; (2) to show how the accuracy of a neural net is modified when human classifications are used to continue training the neural net and (3) to compare a new type of neural net with the back propagation neural network that will be used in the operator experiments.

The specific objectives were: (1) To develop experiment software for an experiment using a back propagation neural network and train the neural network on small but realistic sounds with background noise; (2) train an operator to recognize sounds where he is told the sound type; real time displays and audio will be used; (3) test the operators classification accuracy (in detecting a signal and then correctly classifying it) when he is not told of the occurrence of or type of signal; (4) test his accuracy when he is prompted by the neural network; and (5) continue to train the neural network starting with the weights that were calculated prior to the interactive experiments but with the classifications that were made by the operator and determine the change in network performance.

We have met these objectives and our results indicate that neural networks can be developed to be of great value as an operator aid. The signals that we have chosen for our experiments represent a significant threat class. We believe that neural network technology will make an operator aware of the presence of these signals on the average much earlier. The early detections in general correspond to detection distances that occur at much greater ranges. The ability to improve the detection performance on these signal types with neural networks can be of great value. There are undoubtedly numerous other classes of signals where these networks will also have significant impact.

The two networks that were investigated are time delay neural network where the transient spectra are recognized. The first is a back propagation network that was derived from general purpose back propagation software that is being used routinely by one of us (Rumelhart) in other research at Stanford University. The software is very flexible. It uses several different kinds of processing units and provides diagnostic information about operations within the network. The availability of this software has speeded the research and allowed us to investigate the back propagation architecture in much more depth during the short period of time that is allotted to Phase I efforts.

The other network is radically different from a back propagation network. It contains no weights and thus multiplication is not required. In principle, its operation is analogous to searching through a decision tree; but in practice there is manipulation of a binary sequence that represents the tree. The theory of the decision surfaces that are realized by this network is now well understood. The theory was further developed during this project and under a concurrent contract from DARPA through ONR to MAXIM. The later effort was awarded as part of the DARPA neural network program. The theory and algorithms are described in some detail in the appendices.

The new understanding of this network that has been arrived at during the Phase I time period has enabled us to design trees that have classification accuracies that appear comparable to the accuracies of back propagation networks. However, there is a major advantage. The tree network so far has proven to be much simpler than the corresponding back propagation network. MAXIM has developed a concept for implementing very large fast networks with relatively simple hardware. It is significantly less complex than the hardware that will be required to implement the back propagation network for sonar signal classification. The new hardware concept is described in the Phase II proposal.

Upon reading this report and related appendices, the reader should be left with the following impressions. (1) The adaptive tree is a relatively simple pattern recognizer that can have near optimum classification accuracies; (2) trees can be implemented in VLSI far more readily than backward propagation networks; (3) tree (binary sequence) training is straightforward and (4) the tree separating surfaces and the relation to the nearest-neighbor decision rule are understood.

One further note about the Phase I technical objectives. In both the Phase I proposal and the DARPA proposal (which was written prior to notification of the Phase I award) MAXIM proposed as a technical objective to investigate the ability of networks to segment overlapping sounds. After notification of the DARPA contract award, the scope of the Phase I effort was changed and the segmentation objective was dropped.

The remainder of this section is a project overview that begins with a short description of the signals and the displays that were used in the experiments. There also are network descriptions and a summary of the experiment results. The overview was written for the reader who is not interested in the technical details. There is more detail in the other sections and in the appendices.

1.1 Sounds

At the outset of this project, it was recognized that there would be little validity in our experimental results unless the network training and experiments were preformed with realistic signals and background noise that were typical of those that are encountered by sonar operators. It was also recognized that during the experiments the subjects should have the essential visual information that is used operationally by Navy sonar operators.

Near the beginning of the project, members of the MAXIM staff visited the Navy Fleet Anti-Submarine Warfare School (ASW), in San Diego to observe operators on ASW simulators. We observed displays and listened to selections of underwater sounds from the library maintained by the Naval Sensors Training Aids Department, NAVSTAD, also in San Diego. The operators receive a significant portion of their information from spectral displays. In particular they rely on a gray (green) scale falling raster display of the audio spectrum and there is a single line display that shows locations of spectral peaks. To be an accurate simulation our experimental design should include the information that is generated by these two display types.

At the time of the writing of the SBIR proposal, we planned on recording a series of underwater sounds in the San Francisco Bay. Later we decided that it would be much better if we used the large and professionally recorded underwater sound library that is maintained and continuously updated by NAVSTAD. Many underwater sound selections that we heard in San Diego will be ideal for our experimental purposes. Some of them were classified. We requested, through channels, several unclassified tapes, which we have received, as well as several classified tapes which we have not received. Unfortunately, the unclassified tapes by themselves are not sufficient for our experiments. Fortunately we were in possession of a sound digital tape that was furnished to us by ONR as a part of our

DARPA neural network program. In many ways the sounds on this tape are ideal for Phase I experiments. They are sounds that are important to detect and frequently it is essential to alert the operator when they occur. Several require great concentration for consistent detection.

All of the sounds are pulse type signals that would be admitted by an active sonar. The pulse widths vary from short to long sounds over a large audio frequency range. The signals were synthesized and mixed with real sea noise. Each signal is presented individually in the training samples and with signal-to-noise ratios that vary over 18 dB. Testing samples were provided with different combinations of signal types, different quantities of signals, and with different signal-to-noise ratios. Table 1 lists pulse parameters of those signals. These sounds are excellent examples for use in illustrating how a neural net might aid a sonar operator when he is detecting critical signals. Therefore we decided to use these sounds in our experiments instead of the sounds we had hoped to obtain from NAVSTAD.

1.2 Experiment Software and Computer Displays

Experiment software has been implemented on a SUN SPARC Workstation 1. In all experiment modes the falling raster of the audio spectrum is displayed. A new raster line is added every .5 seconds which is about the same rate used operationally by the Navy. Audio synchronized to the falling raster is used in all modes.

In one mode the subject can select which sound type he wishes to hear; this mode is used so that the subject can learn to recognize each sound type. In a second mode the subject does not know the sound type before presentation, but he can enter his classification and be told the correct answer. This mode is used for self-testing. Learning to recognize the sounds and self testing typically takes about one hour.

SIGNAL	FREQUENCY IN Hz	PULSE LENGTH	REMARKS
A	3515	10 msec	SINGLE CHIRP
B	4687	30 msec (5 msec each pulse)	DOUBLE PULSE
C	3000	10 msec	SHORT BEEP SINGLE TONE
D	3000	100 msec	SLIGHTLY LONGER BEEP SINGLE TONE
E	150	1 sec	LOW FREQ. SINGLE TONE
F	250	8 sec	LONG LOW FREQ. SINGLE TONE

Table 1. Signal Category Parameters

In a third mode the sounds are presented in random order where the types are not told to the subject, and he enters his classification. This mode is used to assess the subjects classification accuracy. In a fou th mode he is alerted and given the classification. These latter two modes are used to measure the accuracy rates during the neural network-operator interaction experiments.

1.3 The Back Propagation Time Delay Neural Network

Both of the neural networks that have been studied are time delay neural networks that operate on the audio time varying spectral energy that is in various audio bands. For example in Figure 1 the spectrum might be divided into n bands and the pattern elements could be x_1 through x_n. Where x_i is the energy in the ith band. The time delayed pattern vector is formed from three successive (in time) spectra, i.e.,

$$\text{PATTERN VECTOR} = \begin{matrix} X_1(1) \\ X_2(1) \\ . \\ . \\ . \\ X_n(1) \\ X_1(2) \\ X_2(2) \\ . \\ . \\ . \\ X_n(2) \\ X_1(3) \\ X_2(3) \\ . \\ . \\ . \\ X_n(3) \end{matrix} \qquad [1]$$

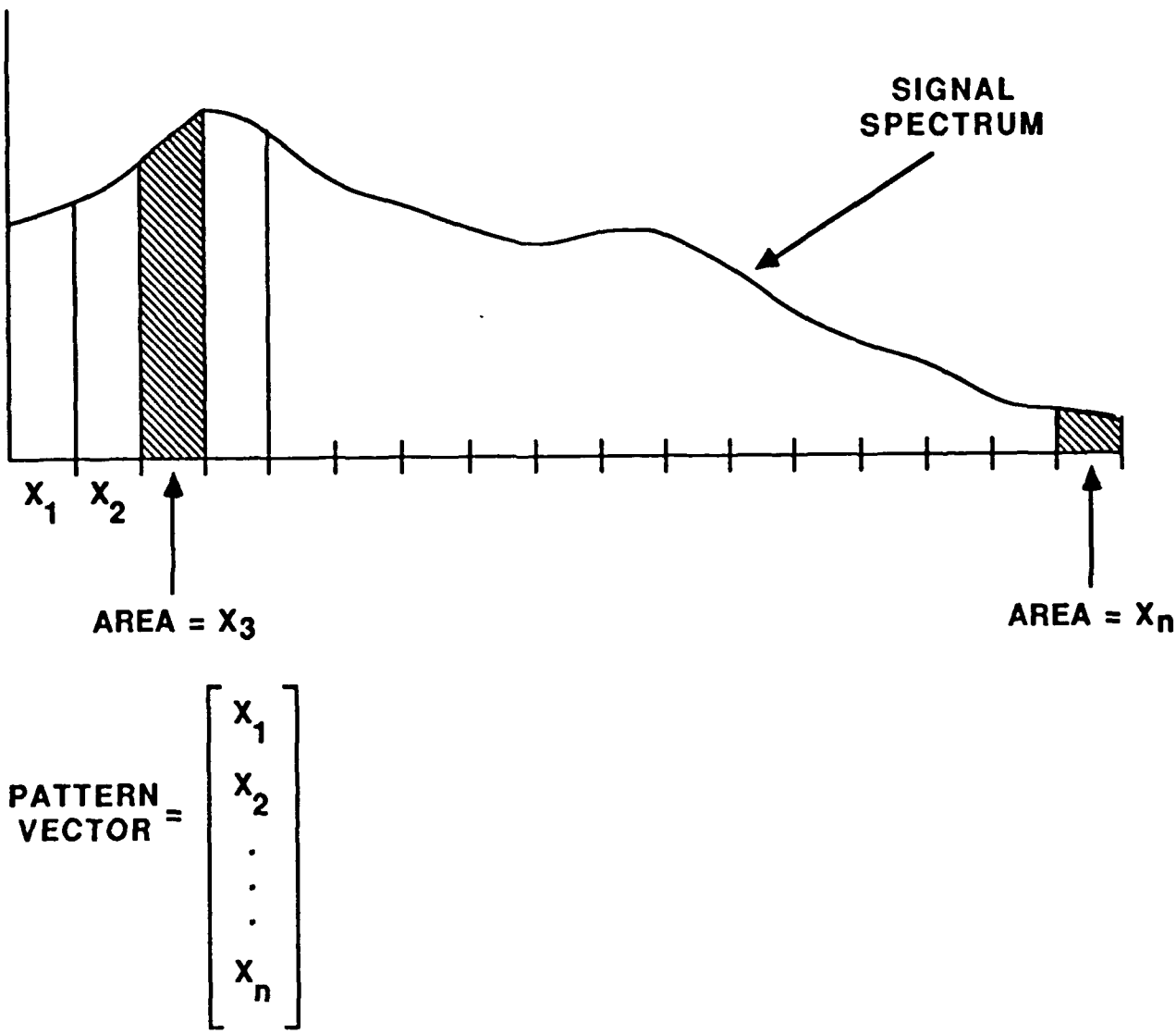


Figure 1. An example of forming the Pattern Vector

With this format, the neural network can learn to recognize the spectral shape and its transients by using the first and second differences of spectral shape. The latter correspond to the first and second time derivative of each element in the spectrum. The input vector to both the back propagation time delay neural network (BPTDNN) that was used in the operator interaction experiments and to the adaptive tree time delay neural network (ATTDNN) has this form.

The BPTDNN is divided into two sections. The first is trained to recognize signals A,B,C and D of Table 1. It has 90 sigmoid processing units in the first layer. They are connected to a time window which is long enough to contain the longest signal, signal D. The second layer contains 5 units which are of the exponential type (this type will be described in more detail in Section II). Each of the 5 is connected to all 90 of the first layer outputs. The output layer also contains 5 units; each is connected to all the units in the second layer. The output units' type is "probability" where the outputs of a fully trained machine are the actual probabilities that the input vector represents the signals A,B,C, and D and noise. This type also will be described in more detail in Section II.

The second section was trained to recognize signals E and F. The window length will contain signal F which is 8 seconds long. The processing unit types in the three layers are identical to the types in the first section. There are 60 units in the first layer, 3 in the second and 3 in the output layer. Again, each unit in a layer is connected to all units in the previous layer.

1.4 Adaptive Tree Time Delay Neural Network

Conventional neural networks contain weights; each weight means a multiplication. In even a moderate sized network, there are tens of thousands of weights. The adaptive tree technology is being developed to make it possible to build neural networks that do not

require multiplication, and thus avoid a major complexity that impedes the construction of truly large and powerful networks on silicon.

Frequently the elements of the pattern vectors are input to neural networks in decimal. Changing the number base for example to binary does not change the statistical properties such as clustering and the overlap of clusters in the pattern vector set. Binary vectors allow the use of Binary logic. Hence, multiplication is not required.

When the input to a neural network is binary, it is convenient to think of the network as a device that implements a switching function. It is well known that AND gate structures in the form of a tree (such as one shown in Fig. 2) can be used to realized any arbitrary switching function. In the example found in Fig. 2, the OR gate will go high when one of the following vectors is present:

$$\begin{pmatrix}0\\0\\0\end{pmatrix}\ \begin{pmatrix}0\\0\\1\end{pmatrix}\ \begin{pmatrix}0\\1\\1\end{pmatrix}\ \begin{pmatrix}1\\0\\1\end{pmatrix}$$

(The reader can verify)

"S" is a positive start pulse that is input after the x_i are in place. When these AND gates are being designed or are being implemented in software, it is convenient to think of the tree networks themselves. The tree equivalent to the AND gate network is shown in Fig. 3. If a 0 is entered into a node, the path moves through the branch from the node to the left (if one exists); if a 1 is entered, the path moves to the right. If a pass through the tree ends at an "end point", the output is "1" otherwise it is "0". We call both the gate network and the tree reduced because they result in an output of "1" before completing a full path. For example, the first two vectors above follow the left most path which is a reduced path. The value at the third node of this reduced path does not affect the result and hence it is a "don't care state".

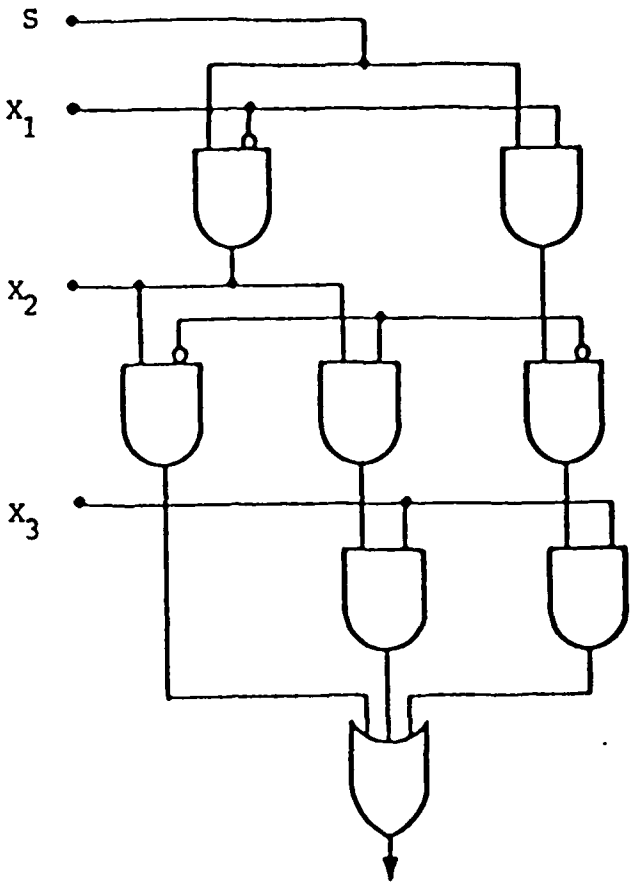


Figure 2. A Reduced "AND" Gate Network

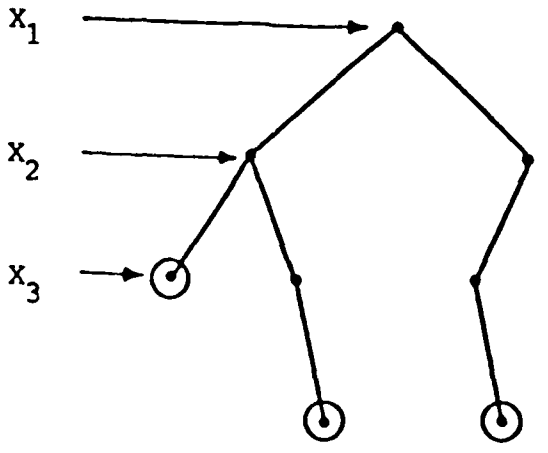


Figure 3. A Tree Represents the AND Gate Network of Figure 2. The x_i are input to the nodes on the ith row. The circled dots are end points that represent AND gate connects to the OR gate. Moving to the left from a node means x_i=0; move right means x_i=1.

The tree can be made adaptive if there is a training algorithm that constructs the tree as it is presented with vectors from the patterns set. Of course the digital logic with the same switching functions can be developed using classical gate minimization techniques. The reason those techniques are rarely used with neural networks is that the calculations required go up exponentially with the input vector dimensions and size of the vector set.

We have developed several tree construction algorithms; each has advantages under different circumstances. They are described in the Appendices. We have tested the adaptive trees on several problems, and our experience has shown that they perform well or better than a back propagation network that is used for the same problem.

During Phase I a set of ATTDNNs has been trained on the same set of training vectors that was used to train the BPTDNN. There is a pair of trees for each of the six signals. One of the trees of the a pair should output a "1" when a corresponding signal pattern vector is input and the other should output a "0". If a noise vector is present the reverse should occur.

The DARPA sound tape contains a training set where the sounds are identified, and a testing set where the sounds are not identified. We have not yet been given the signal locations and types in the testing waveforms so that we cannot measure the classification error rates on the testing set. However, the classification accuracy of the adaptive trees on the training set is close to the back propagation network accuracy and in preliminary tests using the testing waveforms there is good agreement between the BPTDNN and the ATTDNN (see Section II).

An advantage of the tree pair for each signal is that there is an additional capability to detect classification errors. For

example, if none of the "signal-present" trees and none of the noise trees go high, a mistake has been made. Of course, there is a mistake if more than one signal-present tree goes high. If any of these states occur, a test is made to determine which path through a tree is closest to the input pattern vector. It can be shown that with the correct internal architecture the network performance, when trained with large sized sample sets, approaches the Bayes optimum error rate with this error correction procedure (see Appendix A).

Understanding the correct architecture (ordering of the pattern vector elements) to give this performance has been a significant accomplishment during Phase I.

1.5 Important Advantage

The BPTDNN will not run in real time on a serial machine with the speed of the SUN SPARKS Station I. Real time operation requires a classification in less than 10 milliseconds. The ATTDNN can perform this operation in microseconds when implemented in hardware.

1.6 Test Results

As expected, the network operator interaction experiment results seem to confirm the need for a sonar aid. Although the human subjects became proficient at identifying single signals in the training set, they had trouble identifying and categorizing multiple signals presented in a series. On these testing samples, the neural net proved to be a much better classifier than the average subject, especially on the short, transient sounds (A,B,C). When the subjects were given the neural net response as additional information, their accuracy was greatly improved in terms of signal classification and signal detection. The number of false alarms also decreased. It should be noted that all experiments assumed an alert operator. If

fatigue was a factor, as is the case of a real sonar operator, accuracy would surely improve by an even greater margin.

The remaining sections of this report will discuss the neural networks, the experiment software and the test results in detail.

SECTION II
SIGNAL PROCESSING AND NEURAL NETWORKS

This section first discusses the signal processing that generate the pattern vectors. The BPTDNN is described next and finally the ATTDNN is presented.

2.0 Signal Pre-Processing

The six pulse types in Table 1 are narrowband signals. Since the signal-to-noise ratio is usually low (it ranged from +3db to -15db in a 5 KHz bandwidth), a somewhat better spectral estimate can be made by pre-filtering the two signals with narrowband digital filters before the FFT and spectral calculations. This filtering is done digitally with 4 pole Butterworth bandpass filters. (See Figure 4). Butterworth filters were chosen because of their flat frequency response, and relatively sharp cut-off. Because the digital equations are iterative a minimum of computational arithmetic is required. A better neural network performance probably can be obtained if the network is told when the energy in the bandpass exceeds a threshold. This was not done in our experiments. However, an envelope detector and variable threshold were built into the vector generation program to help exactly locate the signal vectors. In Phase II we will experiment with network training using the threshold value as an input. Table 2 gives the filter parameters.

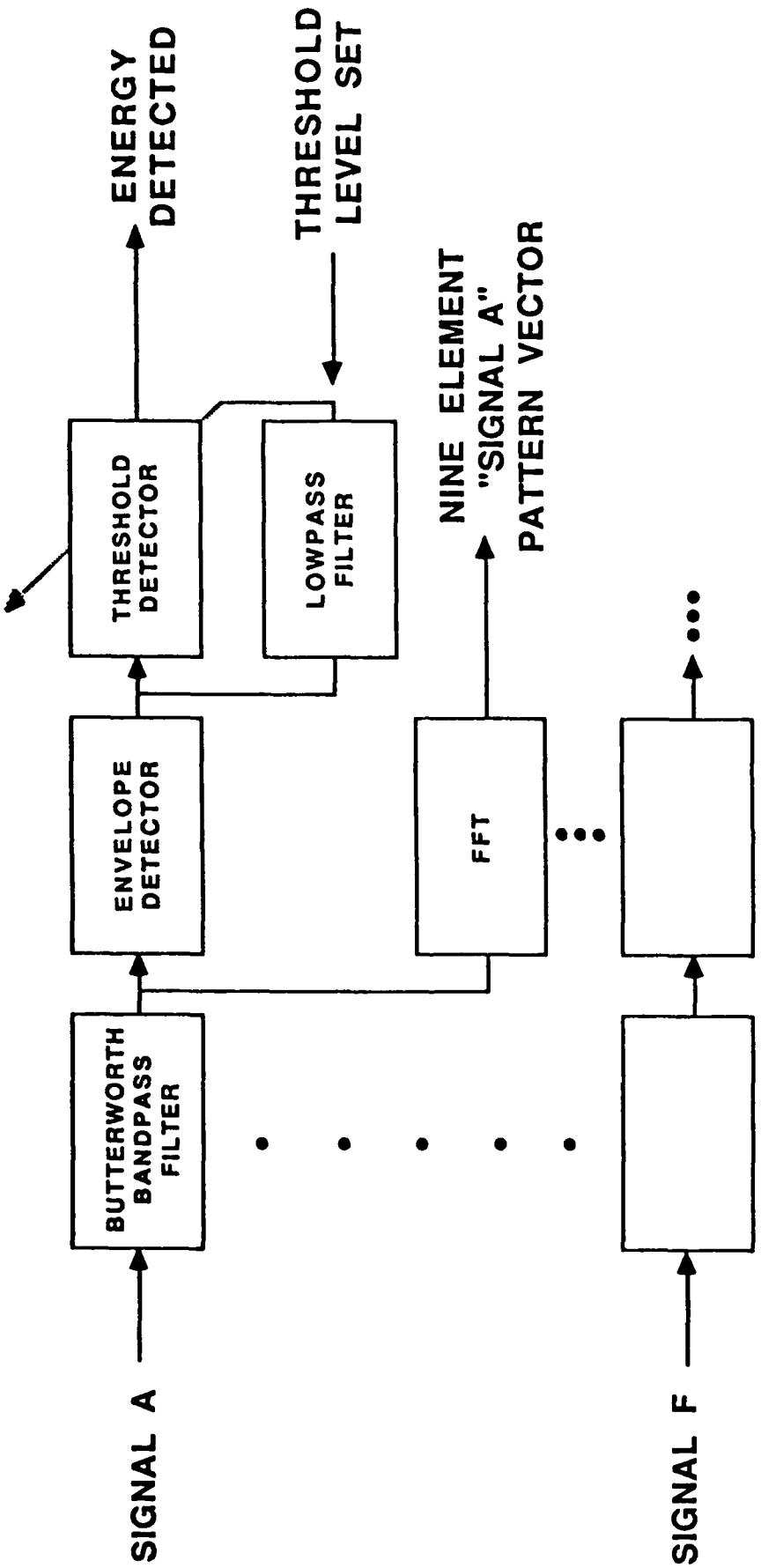


Figure 4 Block Diagram of the Preprocessing Program.

(FOUR-POLE BUTTERWORTH DIGITAL BANDPASS FILTERS)

FILTER PARAMETERS AND PATTERN VECTOR ELEMENTS

SIGNAL	CENTER FREQUENCY	BANDWIDTH	NUMBER OF SPECTRAL ELEMENTS
A	3515 Hz	600 Hz	9
B	4687 Hz	600 Hz	9
C AND D	3000 Hz	500 Hz	7
E	150 Hz	10 Hz	6
F	250 Hz	10 Hz	6
			37

Table 2. Narrowband Filter Parameters

The output of each narrowband filter is transformed with the FFT, and spectral points within the bandpass are calculated. Table 2 shows the number of points from each filter. There are a total of 25 points for the high frequency signals (A, B, C, and D) and 12 for the low frequency signals (E and F). With the high frequency signals the FFT is calculated every 10 milliseconds over a 10 millisecond time interval. For signals E and F, the FFT is calculated every one second over a one second time interval.

2.1 The BPTDNN

The block diagram of the BPTDNN that has been developed for the phase I study is shown in Figure 5. This configuration evolved from experiments with several other configurations. We believe that the detection accuracy is as good as that of a human, when the human is listening intently, and when the signals are those that were described earlier.

Note that the above back propagation network was generated with a unique software tool originally developed by Rumelhart and adapted by MAXIM for the sonar work. The software allows the user to easily and completely define any network architecture including the ability to vary:

- Unit types (sigmoidal, exponential, gaussian, and probablistic)
- Number of layers
- Number of units within layers
- Interconnections
- Initial weights

o Presentation formats and logging capabilities

o Input vector formats

An important feature of the software is that the output is expressed as probability showing the likelihood that an event/ detection has occurred.

If MAXIM is awarded a Phase II contract, the network will be expanded to recognize many more signal types. The expansion will be straight forward; additional sections will be added for each new signal class. When there are sounds that occur in sequence and the sequence determines the signal type (e.g. torpedo sounds), a second set of networks in cascade with the pattern recognition networks will be added to recognize sequences. Also during Phase II, we will experiment with prompts to the operator to look for an up-coming sequence which the neural network expects based on previous sequences.

As was mentioned in the previous sections, time delay is used in both back propagation and adaptive tree networks. The time delay is introduced through the use of the windows such as those shown in Figure 5. Recall that there are 25 spectral components that represent the high frequency signals. The components are introduced into the left side of the left hand window every 10 milliseconds(a new 25 dimensional vector is entered) and each vector in the window is shifted over one position.

Each of the 90 S-units in the input layer is connected to three of the vectors in the window. For example all 10 units in the first column of the first layer are connected to vectors 1, 2, and 3 in the window. The 10 units in the second column are connected to the vectors 2, 3 and 4 etc. Because a processing unit is connected

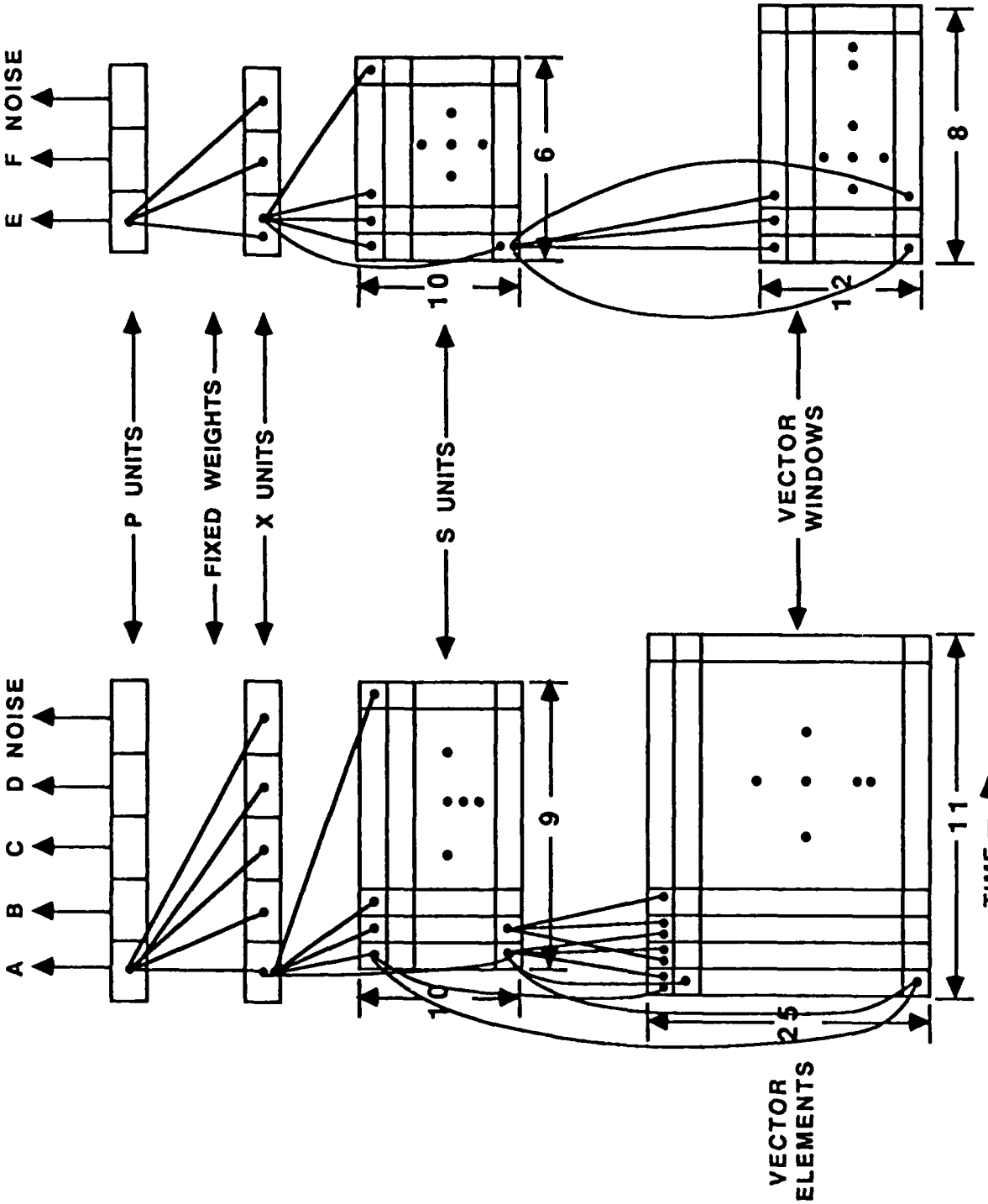


Figure 5 Block Diagram of the Back Propagation Network.

to three successive vectors input to the unit has the form of Eq (1). There is a similar connection scheme for the low frequency signal window.

Each of the 5 X-units is connected to all 90 units in the first layer. The X-units (X for exponential) are used because there is a significant reduction in the number of units that are required in the second layer, and there is a reduction in the number of training cycles that are required. The X-units are the same as the S-units except for the non-linear function (called the squashing function in S-units). In both units the sum

$$net_i = \sum_j w_{ji} x_j - b_i$$

is calculated where w_{ji} is the jth weight of the ith unit, x_j is the output of the jth unit in the previous layer (or the jth element of the input vector when the ith unit is in the first layer) and b_i is the bias term. In the X-unit the non-linear term is

$$f(net_i) = e^{net_i}$$

This function when used with the P-units in the output layer allows estimation of the probabilities of each signal for the combination of vectors that are in the input window. There is a version of the back propagation training rule where X and S-units can be intermixed in any order.

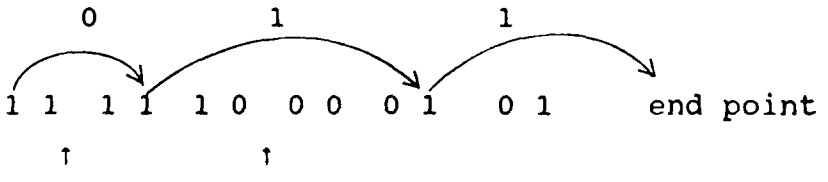
The P-units have fixed weights (nominally set equal to one). The weighted inputs from the X-units are summed to form the P-unit outputs; there are no non-linearities. The goal is 1 for the P-unit output that corresponds to the signal class of the input vector and 0 for all other outputs. With these goals the network output will

converge to the probabilities that the input vector is from each of the signal classes or due to noise. The classification is determined by which probability is highest. With sufficiently large training sets this classification will be the same as the Bayes decision. Use of the networks ability to estimate probabilities will be explored in Phase II where the operator may be advised to take a course of action based on the probabilities and not just on whether or not a signal is classified.

2.2 ATTDNN

Appendices A and B, which described the tree and binary sequence training in great detail, are included in this report because few know about the trees outside of MAXIM and Stanford University. We have used them in several areas such as signal processing, character recognition and of course sonar signal classification. We have found them to have accuracy rates that are comparable to the back propagation neural networks that were used on the same problem. But, the trees have always been significantly less complex and considerably faster.

This is true for the trees that have been developed in Phase I. The adaptive tree when implemented in hardware can be constructed as an AND gate network,(as in Fig. 2). Special purpose digital logic can be used to realize a tree where the tree is represented by a comma free binary sequence; two bits are required for each node. For example, the binary sequence for the tree in Fig. 3 is



Each binary pair represents a node. The digits 1 1 mean branches both to the left and to the right; 1 0 means a single branch to the left; 0 1 represents a single branch to the right, and 0 0 means there are no branches (i.e., an end point). The number of pairs in layer n+1 equals the number of 1s (branches) in layer n so the pair at the end of a layer can be found and no commas are required. The curved arrows trace the path when the vector is input. The first arrow leaves from the first 1 (which represents the zero branch) and goes to the first pair in the second layer. The second arrow leaves that pair from the second 1 (which represents the 1 branch) and goes to the second pair in the third layer. (If an arrow leaves from the ith 1 in the row, it goes to the ith pair in the next row). Since the last bit in the vector is a 1, the third arrow leaves from the 1 of the second pair in the third row for the end point.

If the last bit had been a 0 the departure would have been from the left bit in the pair which is 0 which means there is no branch. Thus the tree output would have been 0.

Software implementation on a serial computer is an emulation of the above process.

Appendices A and B describe the algorithms for building the binary sequences. Training vectors can be presented randomly or sequentially. If the presentation is random, the trees can track a changing environment. A sequential presentation requires only three passes through the training vector set.

Appendix A discusses element ordering in the binary vector. For example there is no electronic or computational need to enter the first bit into the first node; it can go in at any level. Re-ordering can result in enormous reduction in the number of nodes;

just as important, it can greatly reduce the size of the training set that is required for training to adequate levels of accuracy.

Appendix A shows that with proper ordering there is a strong relationship between the adaptive trees and the nearest neighbor decision rule. But the trees can be significantly more efficient. In general the elements that have least variability over a signal class and which are most significant (i.e., represent the larger powers of 2 as they come from the A/D converter) are entered high in the tree.

Section I noted that the adaptive trees developed for the sonar data classification are in a two tree configuration shown in Fig. 6. Specifically, the figure shows the configuration for recognizing signal D. The window is eleven columns wide which will enclose all of signal D. Recall that only seven of the twenty-five high frequency spectral components came from the D filter, and only these seven are input to the window. Notice that the column connections in Fig. 6 differ from the column connections shown in Fig. 5. Every element in all eleven vectors is input to the trees. After the floating point to binary conversion the binary vector is reordered and input to the tree pair.

Table 3 is a computer printout that summarizes the Signal D tree pair parameters.

The top group of numbers gives the reordering of the three hundred and eighty-five elements of the binary pattern vector. For example, element eight is entered into the first node, element twenty is entered into the second layer, thirty-three into the third layer, etc. There are sixty-three nodes in the D Tree and sixty-five in the not D Tree. The number of nodes for the six tree pairs range from fifty to one hundred and twenty-eight.

ADAPTIVE TREE STRUCTURE FOR RECOGNIZING SIGNAL D

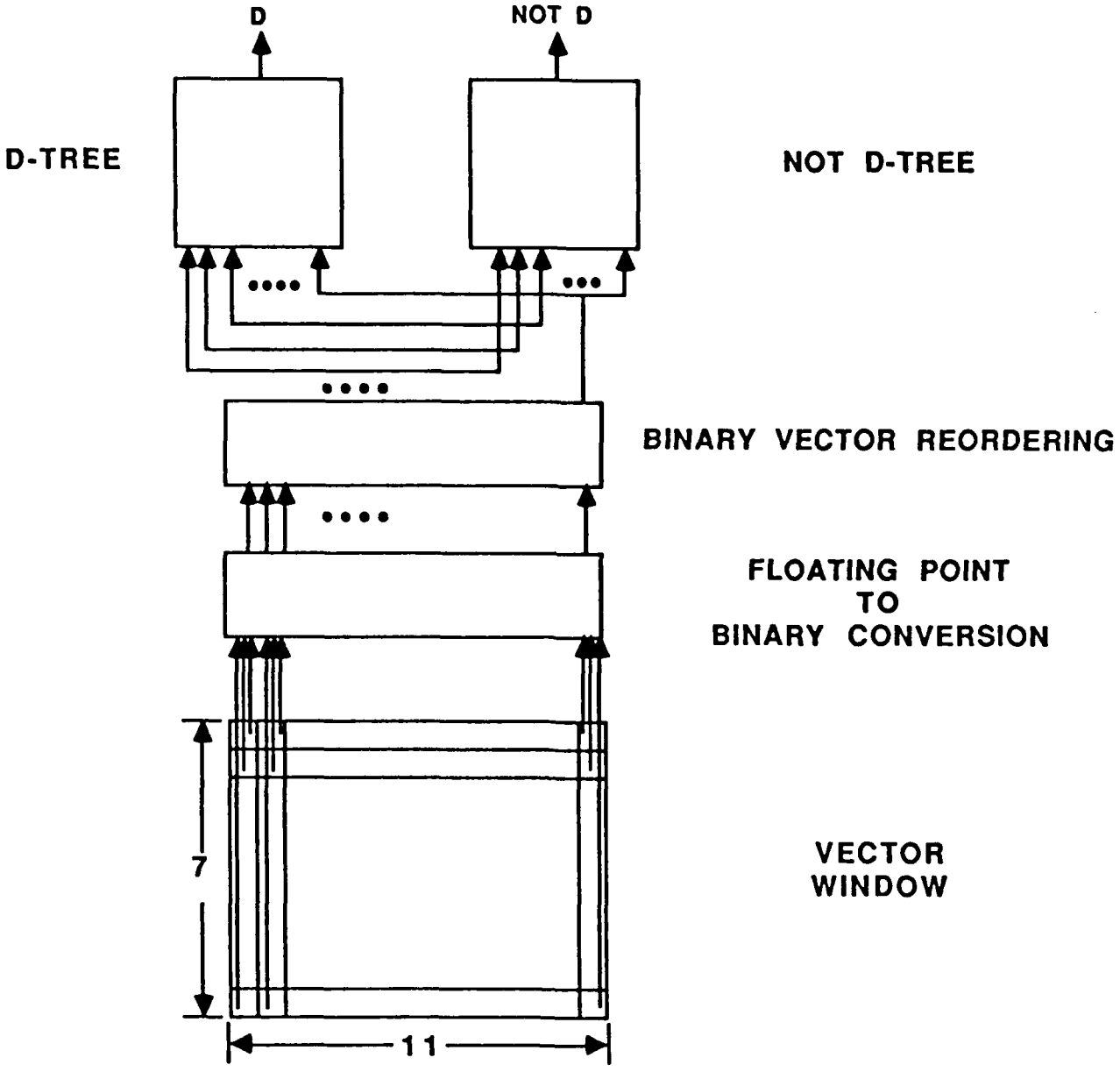


Figure 6 Block Diagram of the Tree Configuration for Signal D.

```
8  20  33  54  55  59  69  73  78  89  90  94  110  124  125  129  154  159
160  164  194  195  199  204  205  225  229  230  234  244  264  265  269  275
288  299  300  304  334  335  339  344  349  354  365  380  130  134  135  139
144  149  49  158  14  15  163  58  165  169  170  179  184  189  193  18  60
198  64  200  19  70  214  219  220  224  9  228  75  23  233  84  235  239
85  245  249  250  254  255  260  88  24  268  25  270  274  93  285  29  295
298  95  99  303  100  305  309  310  314  320  324  329  330  333  104  30
338  114  340  119  345  10  350  40  355  359  128  45  120  50  35  39  74
175  4  180  44  240  315  185  79  325  190  80  140  105  145  109  150  5
155  210  34  215  280  115  65  360  289  290  385  76  77  37  166  167  168
38  1  171  172  173  174  81  176  177  178  82  83  181  182  183  2  41
186  187  188  86  87  191  192  42  43  21  196  197  91  92  22  201  202
203  46  47  206  207  208  209  96  211  212  213  97  98  216  217  218  48
11  221  222  223  101  102  226  227  103  12  51  231  232  106  107  108
236  237  238  52  53  241  242  243  111  112  246  247  248  113  13  251
252  253  26  116  256  257  258  259  117  261  262  263  118  56  266  267
57  121  122  271  272  273  123  27  276  277  278  279  28  281  282  283
284  126  286  287  127  6  61  291  292  293  294  62  296  297  131  132
133  301  302  63  7  136  306  307  308  137  138  311  312  313  31  66  316
317  318  319  141  321  322  323  142  143  326  327  328  67  68  331  332
146  147  148  336  337  32  16  151  341  342  343  152  153  346  347  348
71  72  351  352  353  156  157  356  357  358  17  3  361  362  363  364  36
366  367  368  369  370  371  372  373  374  375  376  377  378  379  161  381
382  383  384  162

1  63
4432  4043  2440  3233  20  2202  2222  2222  2227  2222  2222  2222  2222
2222  2222  2300

2  65
4422  4042  2420  2223  403  2007  2224  2022  2222  2222  2222  2222  2222
2222  2224  2072  0


The total number of nodes is 128
```

Table 3. Summary of Signal D Tree Pair Parameters

2.3 Performance

The trees were trained to give errorless performance on the training set of signal and noise vectors. Because of the small number of signal vectors in the training set that was furnished by DARPA, it was not considered meaningful to divide the signals (which were furnished with the classification for each signal) into a set for training the trees and another for testing the trees. However, there are ample noise samples in the DARPA furnished training set which can be divided into vectors to train the trees and vectors to test the trees for noise recognition. The error rate in the classification of the noise vectors averaged over the six tree pairs, is very small. Hence, we were not able to accurately estimate the noise misclassification rate. The number of false alarms is negligible. This is also true for the BPTDNN. Occasionally the ATTDNN would not classify noise vectors as noise. The indication was "unclassified", however, it virtually never confused noise with signal. Judging from our results with the training set, both the BTTDNN and the ATTDNN will have adequate error performances for sonar operator prompting when the signals are active pulses.

The following are some initial speed estimates. The back propagation network takes approximately .2 sec. per classification on the SUN SPARC Workstation 1. To classify in real time it must make a classification in 5-10 msec. for signals A, B, C, and D. The adaptive tree network however will make a classification in under 5 msec. on the SUN SPARK Station 1. We conclude that the back propagation network for recognizing these six signals will not operate in real time on a SUN SPARC Workstation 1 class serial machine when the program is written in a higher level language such as C. However, the trees (at least for these six signals) will run in real time on this class of a machine.

If the back propagation network is used, special purpose hardware will be required. In preparing for the Phase II proposal writing, MAXIM developed a concept for adaptive tree hardware that we consider a breakthrough. It will be possible to realize very large trees with hardware that is significantly less complex than the hardware that will be required to implement the equivalent back propagation neural network. This concept is explained in detail in the proposal.

2.4 Human Interface

The human interface software was developed as a tool to explore the sonar operator/neural net interaction in terms of accuracy. One major focus of Phase 1 was to verify the need for a neural network sonar aid and obtain preliminary findings in terms of accuracy. The software incorporates auditory and visual information about the signal set. The operator listens to the actual audio sample while watching a falling raster display of the magnitude of the associated spectrum. Again, the falling raster display was modeled after the sonar simulators seen in San Diego at the ASW school.

The human interface software is run on a SUN SPARC Workstation 1 under the UNIX operating system as are all the neural network models. The interface was written in "C" for versatility and makes use of standard graphics software provided by SUN. It is operated with a mouse using the point-and-click method.

The interface software is made up of three modules; the trainer, the tester and the netprompt tester. A detailed description of each module is given below. Each module is run for different phases of the experiment. The data flow is identical in all three modules and is shown in Figure 7. The digitized audio samples (provided by ONR) are converted to analog by an A/D internal to the

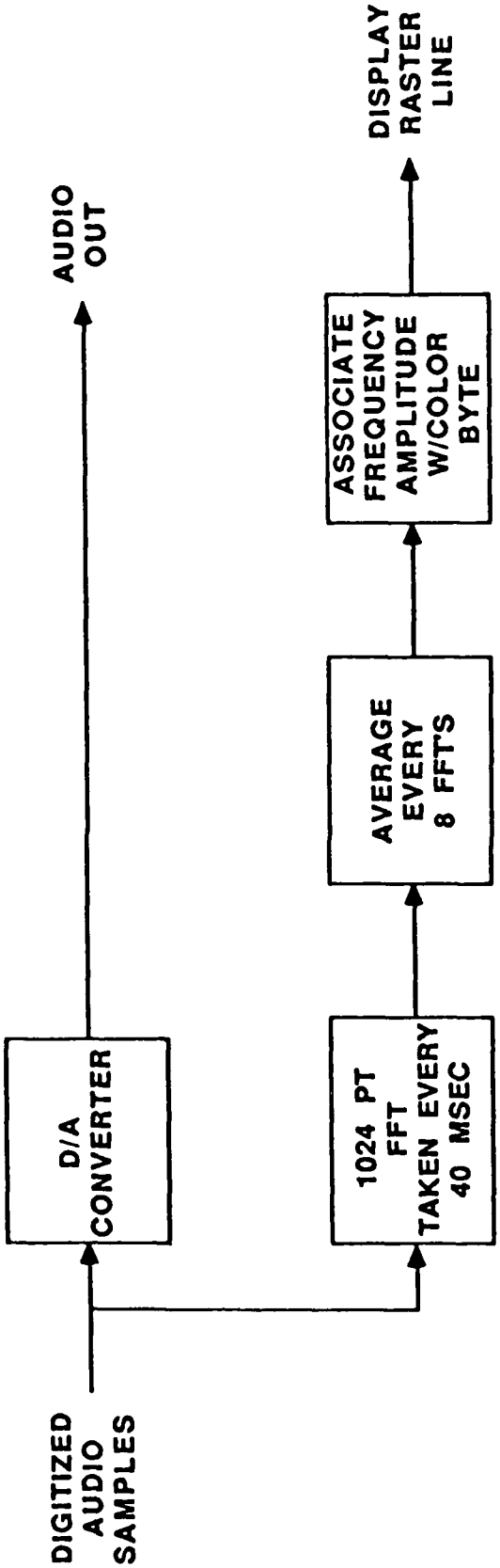


Figure 7 Data Processing - User Interface Software.

SPARC and sent out the audio port to a pair of headphones. Simultaneously, a 1024 pt FFT is taken from the digitized samples every 40 msec. The square of the magnitude is taken of the resulting 513 spectral points. Every eight FFT's are stored and averaged. Every point in the averaged FFT is then associated with one of six color bytes according to magnitude. These 513 color bytes make up one raster line which is displayed in time with the associated audio.

2.5 Trainer Module

This module aids in training the human subjects to recognize the six different sound categories (A-F). Figure 8 pictures the display window of the training module.

The upper left-hand corner of the window is the main menu which displays all control functions. The user clicks on the sound category and example number of the signal which he or she wishes to train on. The user then clicks on the play button to hear the audio and watch the associated falling raster display. Each example of each category contains only one signal. The training example will play over and over again until the user clicks on the stop button. At this time the user can pick another training example and start the process over again. A sliding volume control is also provided to the user. This operations method allows the user to train at his or her own pace, repeating those examples which where difficult to recognize.

The main menu also features a self-test option in which the user can test himself on the training samples to judge how well he is learning the sound categories. The user clicks on the test option and then clicks on play. A random example is chosen from one of the categories. The user categorizes the sample and clicks on the appropriate category letter (A-F). The sample keeps playing until

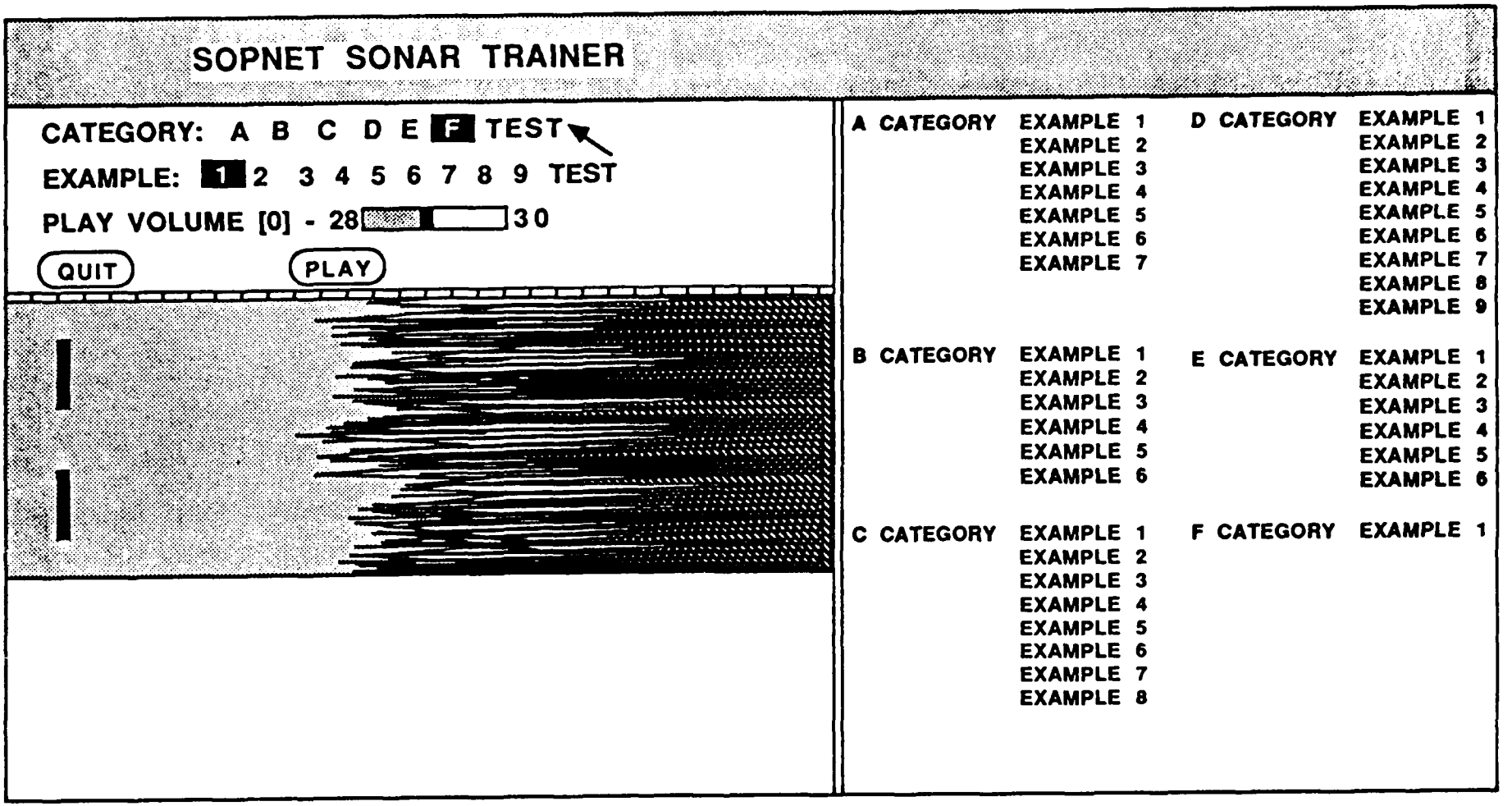


Figure 8 Training Window Display.

the user clicks on the correct answer. This process can be repeated over and over again.

The right-hand side of the display window is a signal reference guide. It displays the six categories and each cataloged example of each category. These are the samples the user can choose from. It should be noted that any catalog of sounds can be interfaced to the software module just by changing a catalog file.

The bottom left-hand side of the window is the falling raster display. Unlike the simulators seen at the ASW school, the display is in color instead of gray-scale which is a definite improvement. However, just like the raster displays used by the Navy, MAXIM's falling raster does not display short, transient sounds. The grain of the raster is just not fine enough to display such sounds. This is a common problem in the sonar community. The only way to detect these sounds is by hearing them. A neural network sonar aid would greatly help the operator detect these transient sounds which are easily missed.

2.6 Tester Module

This module administers tests to the operator using a similar format to the trainer module. Figure 9 pictures the display window of the tester module. The only real difference is in the main menu. Instead of having category and example choices, only the number of the current test being played is displayed. The user plays through all the tests, writing down on an answer sheet the category (A-F) of each sound seen and/or heard. Again, the tester software module can be interfaced to any set of test samples just by editing the catalog file.

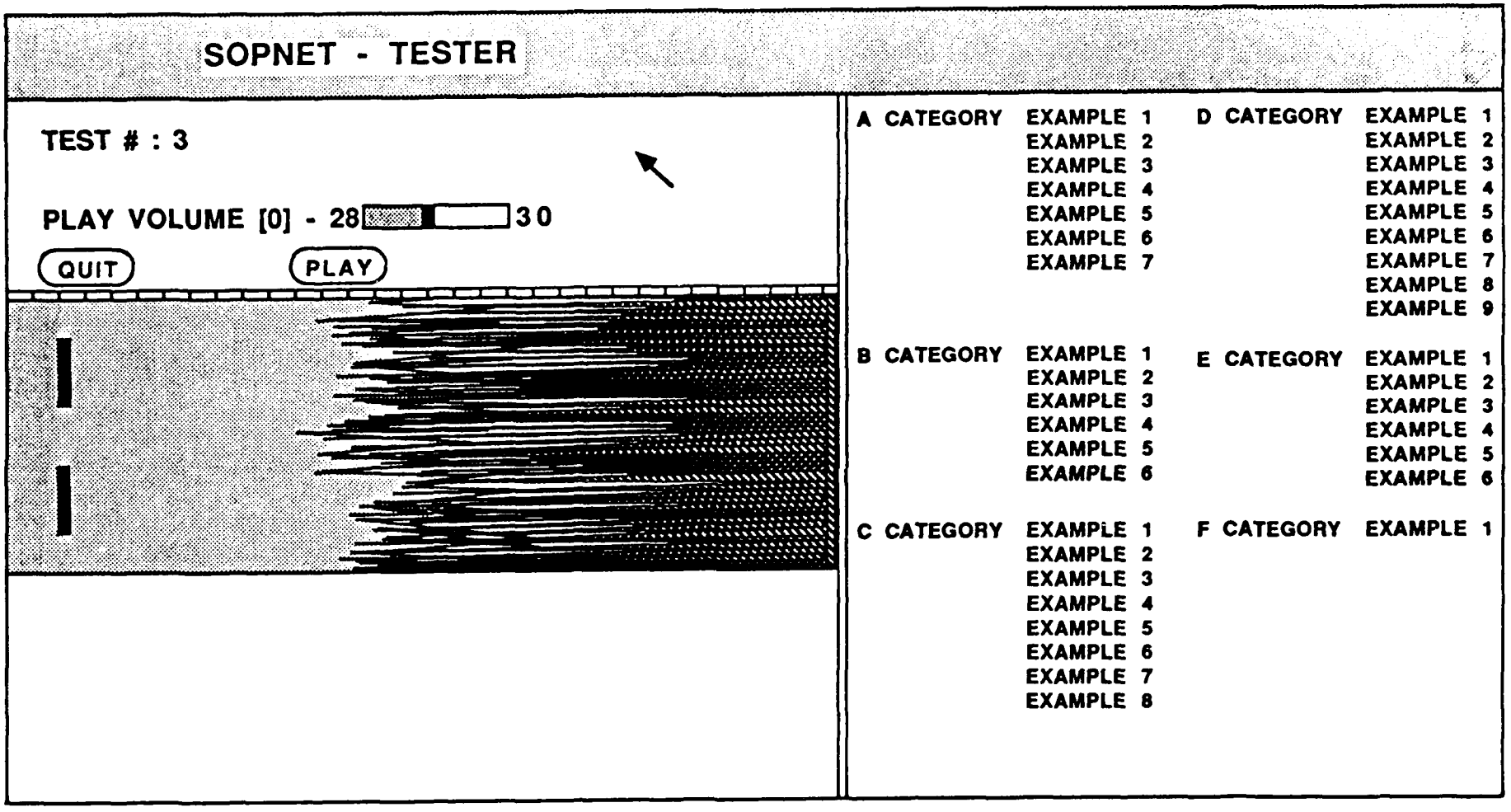


Figure 9 Testing Window Display.

2.7 Netprompt Tester Module

The final module not only displays the test number but also a category prompt obtained from the neural net results. The module display is pictured in Figure 10. As a testing sample plays, the operator is prompted by a category letter which disappears after a few seconds. There is a slight delay between hearing the actual sound and the category prompt. It should be noted at this point that the network is not running in real-time. All testing files have been pre-processed to incorporate the neural net response.

2.8 Operator - TDNN Interaction Experiments

The objective of the TDNN interaction experiments was to obtain preliminary results concerning the feasibility of a sonar aid. Eight co-workers with varying levels of perception skills were used in these experiments. In Phase II, detailed experiments will be conducted with a more select group of subjects. Pre-testing will be done on all novice listeners to screen out subjects with poor perception skills. Experiments will also be performed on a group of "experts". A greater number of subjects (30-50) will also be used to obtain more meaningful statistics in Phase II.

The actual tests consist of temporal patterns of all six signal types on background sea noise. Each test can contain any number of signals in any order at varying ratios of signal-to-noise. Not all signal categories are contained in each test. The operator must truly listen to the test sample since there are no preconceived notions concerning number of signals and types to be contained in each test file.

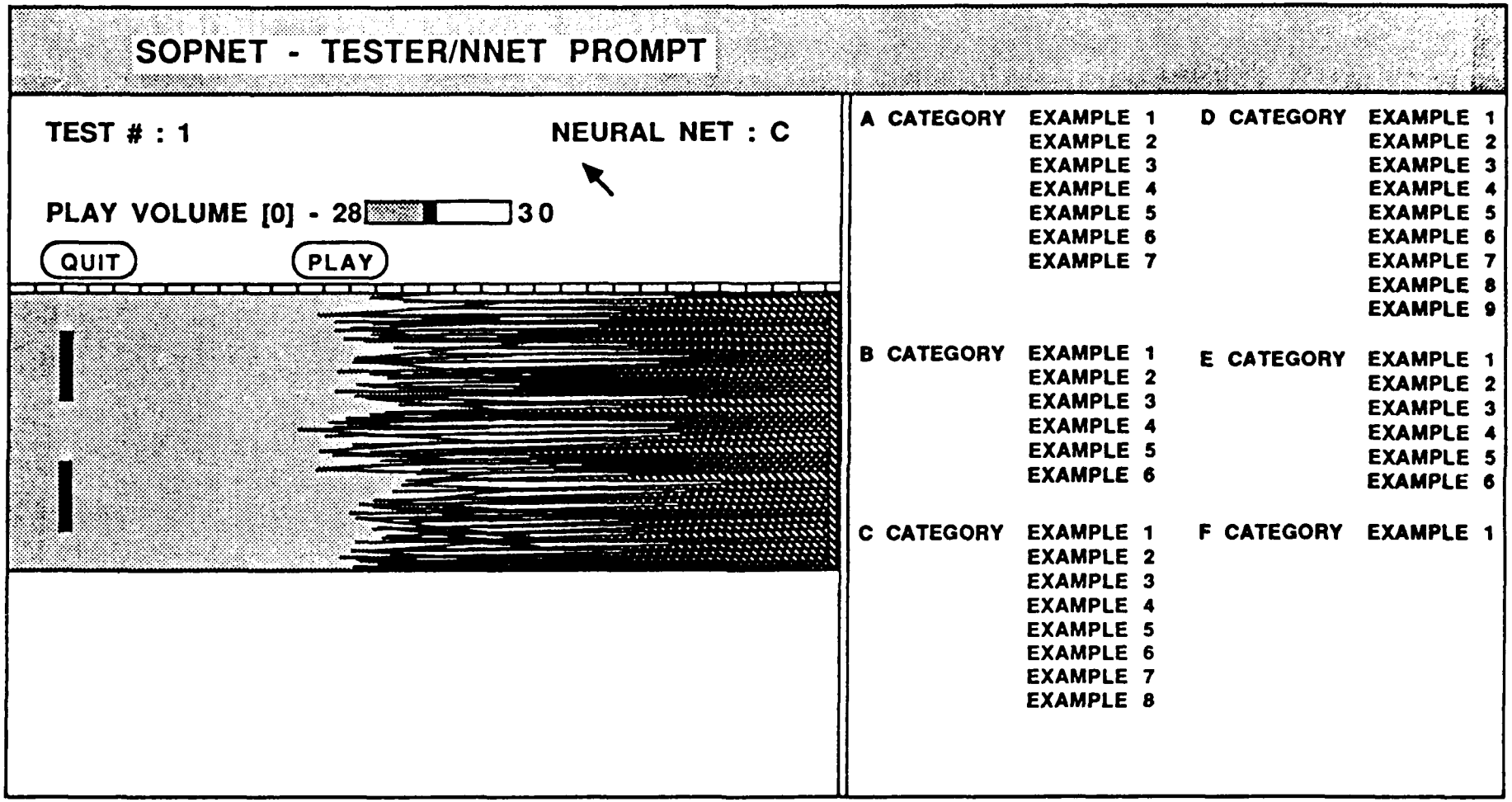


Figure 10 Testing Display With Neural Net Prompt.

2.9 Experiment #1 - Operator Baseline

The testing files, as well as the training files were provided by ONR under the DARPA related contract. An actual accounting of signal placement and type was not provided with the testing files. Using narrow-band filters, threshold detectors and advanced signal processing techniques, MAXIM feels with a high degree of confidence that a subset of these testing files has been accurately classified. The accuracy of the back propagation network on this subset of test files was 100%. This statistic will be used as a baseline in calculating human performance. We know that 100% accuracy will not be found on all files and can not be expected in real world applications, especially when large amounts of noise are present.

2.10 Procedure

Before beginning any experiments, the test subject was asked to train him or herself using the trainer module. This training process took on an average of a hour. The subject was then asked to use the self-test mode on the training set signals to insure competency. The average user could classify the single training signals with a 90% accuracy.

Now adequately trained, the subject began the first set of tests using the tester module. The subject listens to 10 test files which contain 3 to 7 signals each and writes down in time sequence the signals heard or seen referring to the falling raster display. Each test file is presented only once since a real sonar operator would not have the luxury of hearing the data over and over again.

2.11 Results

The results of this experiment are shown in the confusion matrix, Table 4. The most difficult signal to detect was signal B,

ACTUAL SIGNAL OCCURANCES	OPERATOR IDENTIFIED SIGNAL OCCURANCES						
	A	B	C	D	E	F	N
A	70%	7%	8%	—	—	—	15%
B	7.5%	67.5%	—	—	—	—	25%
C	4.3%	—	95.7%	—	—	—	—
D	—	—	4%	96%	—	—	—
E	—	—	—	—	100%	—	—
F	—	—	—	—	—	100%	—

Table 4. Confusion Matrix: Human Operator Alone

the double chirp. In fact, most subjects were unable to distinguish it from noise much less classify it correctly. There was also broad confusion with all the high frequency signal categories when classifying signals A and C. As expected, the shorter transient sounds were the most difficult to detect and categorize. However, all subjects were quite accurate when classifying the longer, lower frequency tones of E and F. This is probably due to the fact that E and F were very visible on the falling raster display.

Not shown by the confusion matrix is the number of false alarms that occurred. On the average, a subject detected 5.4 signals (44 were presented) that were not actually there. It seems the subjects were concentrating so hard to hear one of the six signals, they classified irregularities in the background sea noise as actual signals. This is a common phenomenon and would probably only increase the longer each subject listened.

2.12 Experiment #2 - Operator Prompted By Neural Net

Experiment #2 explores the accuracy of an operator when a category prompt is given. An inverse of this scenario would be the operator prompting or "re-training" the neural net. However, the existing neural net software does not run in real-time. To even come close to real-time training, the net would only see the new example classified by the operator once. Usually, the net needs to see training examples time and time again before the weights are altered enough to produce a correct output. Thus, this type of training would only confuse the back propagation network and significantly decrease its accuracy. Therefore, we will only study the effects of the net on the operator and not the effects of the operator on the net.

2.13 Procedure

In this experiment, the same ten test files were pre-processed to include the category decisions made by the back propagation network. As the test data was being presented a category prompt was interactively being displayed for a few seconds. There is a slight delay from hearing/seeing the actual signal to seeing the prompt. The subject was instructed to use the auditory information, falling raster information and the neural network prompt to come to category decisions. The subjects were also led to believe that the neural net response was not always correct in an attempt to keep the subjects from using the net response as a crutch.

2.14 Results

The results of this experiment are shown in another confusion matrix, Table 5. There is a remarkable improvement in signal detection and classification. The most obvious example is the case of signal B. Signal classification was improved by over 30% and signal detection was improved by 20%. Signal classification was improved by over 37% in the case of signals C and D.

The number of false alarms was also decreased. On the average, a subject only falsely detected 3.6 signals (out of 44) in the noise as opposed to 5.4.

2.15 General Conclusions

As expected, the above results seem to confirm the need for a sonar aid. Although the human subjects became proficient at identifying single signals in the training set, they had trouble identifying and categorizing multiple signals presented in a series.

ACTUAL SIGNAL OCCURANCES	OPERATOR IDENTIFIED SIGNAL OCCURANCES						
	A	B	C	D	E	F	N
A	51%	14%	15%	5%	—	—	15%
B	17%	36%	2%	—	—	—	45%
C	19%	3%	58%	15%	—	—	5%
D	3%	—	32%	58%	—	—	7%
E	—	—	—	—	100%	—	—
F	—	—	—	—	—	100%	—

Table 5. Confusion Matrix: Human Operator With Neural Net Prompt

On these testing samples, the neural net proved to be a much better classifier than the average subject, especially on the short, transient sounds (A,B,C). When the subjects were given the neural net response as additional information, their accuracy was greatly improved in terms of signal classification and signal detection. The number of false alarms also decreased.

It should be noted that operators in both experiments were assumed to be alert and fully concentrating. If fatigue was a factor, as is the case of a real sonar operator, accuracy would surely improve by an even greater margin.

APPENDIX A

TWO TRAINING ALGORITHMS AND ERROR CORRECTION

We have developed several training algorithms; each has advantages under different circumstances. Here, we will describe the two both of which have been tested during Phase I for sonar signal classification.

Algorithm 2 is significantly more complex so complete examples of tree building and binary sequence construction using this algorithm are given in Appendix B.

In writing Appendix A we hope to leave the reader with the following impressions.

(1) The adaptive tree is a relataively simple pattern recognizer that can have near optimum classification accuracies; (2) trees can be implemented in VLSI far more readily than backward propagation networks; (3) tree (binary sequence) training is straightforward and (4) the tree separating surfaces and the relation to the nearest-neighbor decision rule are understood.

ALGORITHM 1.

During training the pattern vectors are presented to the tree one at a time. The algorithm is a set of three rules for adding branches to existing nodes and for establishing new nodes.

The vector is entered into the tree and a path is followed until a node is reached that has no branches or there is not a branch in the correct direction (e.g., $x_i = 0$ but there is no branch to the left). The following rules then apply:

- If there are no branches from the node, no branch node can be added until a category 0 path arrives at the node (the node is activated).

- If a category 1 path arrives at an activated node and there is not a branch to continue the path in the correct direction a branch is added in that direction.

- Branches are not added to a node that is not activated, and there are additions only on the arrival of category 1 paths.

After training, the tree can be used for classification. The output is a one if the path arrives at a node with no branches (an end point) or a zero if the path arrives at a node with one branch which is in the wrong direction.

Figure A1 illustrates the algorithm with the XOR problem. Here the output should be zero when the input vector is

$$\begin{pmatrix}0\\0\end{pmatrix} \quad \text{or} \quad \begin{pmatrix}1\\1\end{pmatrix}$$

and one when the vector is

$$\begin{pmatrix}0\\1\end{pmatrix} \quad \text{or} \quad \begin{pmatrix}1\\0\end{pmatrix}$$

This network is much simpler than the back propagation network of the same problem.

Another version of this algorithm activates the node only when the category 0 probability at the node exceeds a threshold, and branching cannot occur until the category 1 probability also exceeds threshold. This version can result in significantly smaller trees when the vector set is "noisy", but the trees will not separate the training set with 100 percent accuracy. However, there is a tight

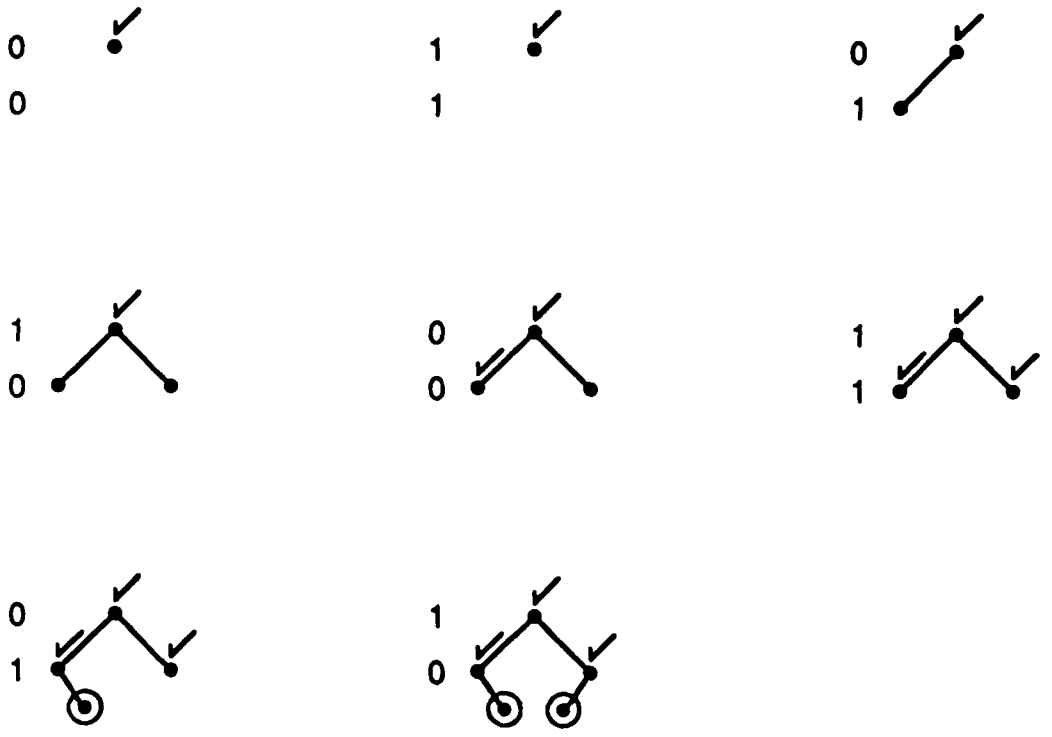


Figure A1. Building the Tree to Solve the XOR Problem. The two digits to the left of a tree are the elements of the input vector. The check marks indicate that the node has been activated.

bound on the error probabilities which is less than the larger of the thresholds.

ALGORITHM 2.

During the first step of the training procedure, all of the category 1 vectors of the training set are input one at a time. Consequently, a tree is built that has paths for each of these category 1 vectors. We will illustrate using the category 1 vectors for the tree of Figure 3 in the main report. Figure A2 shows the training procedure. The three binary digits to the right of trees a through h are vector inputs. The first input is

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

so the first path is that of tree a. The second input is

$$\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

so only one branch is added to give tree b.

After all four category 1 vectors have been input, tree d results. If the four bottom nodes are considered end points, this tree will give the correct output, however, it will not have the minimum number of nodes. The minimum tree is obtained in the second step of the training procedure by a process called tree pruning. The set of category 0 training vectors is input and the paths are traced out through the tree. Each node that is entered by a category 0 path is noted (the check marks in trees e through h). In tree e the path is from node 1 to 2 to 5 and out of 5. Thus nodes 1, 2, and 5 are checked. After all four category 0 vectors are input, nodes 1, 2, 3, 5, and 6 are checked.

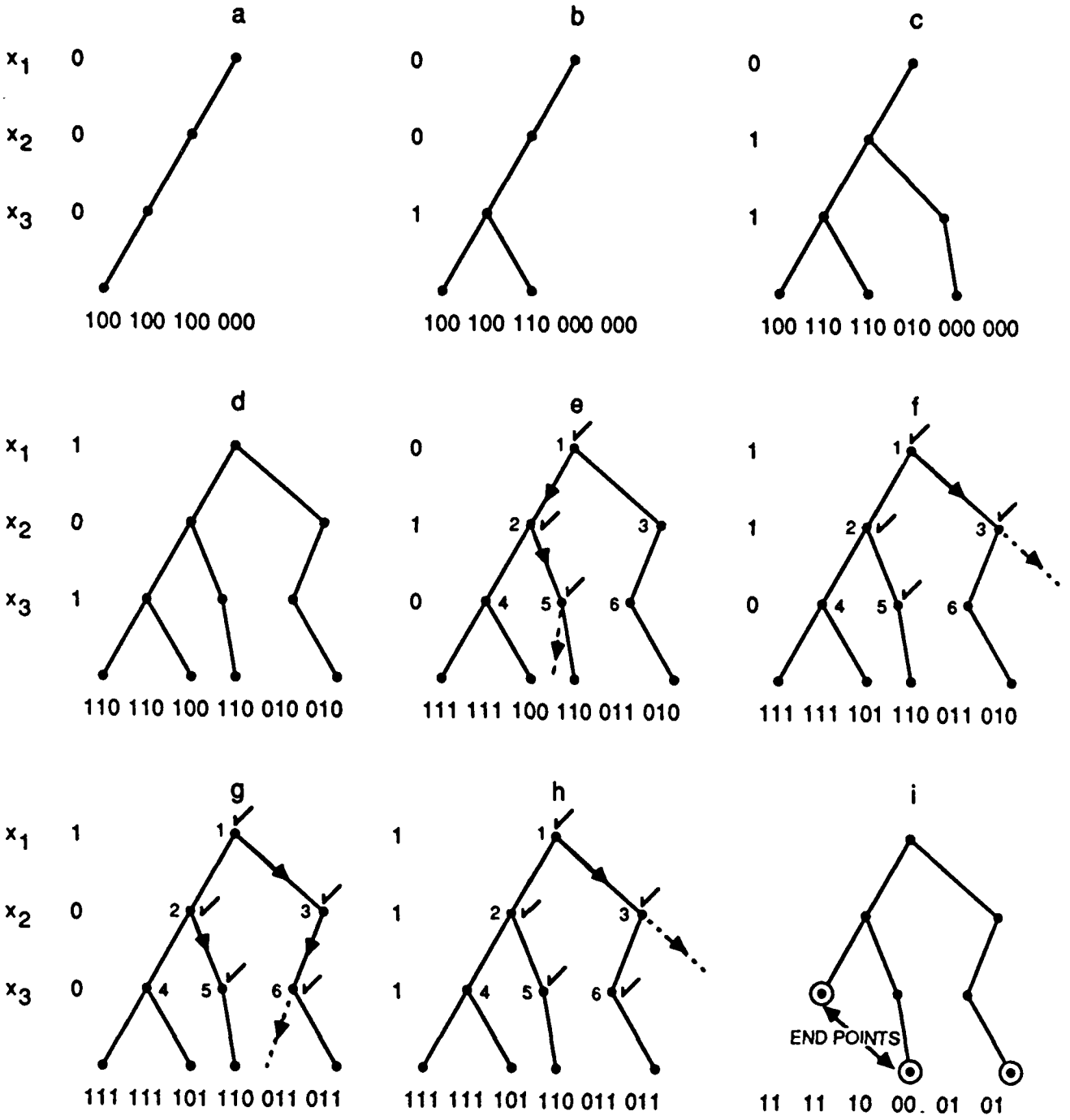


Figure A2. The Sequence of Trees Developed During Training. The first four are from step 1 and the second four are from step 2.

Branches are pruned by the following rule:

- If an unchecked node is connected to a checked node, the former is converted to an end point and the nodes and branches of all category 1 paths that pass through the first unchecked node are eliminated.

This can be done by inputting the category 1 training vector set. When a path first arrives at an unchecked node, this node is designated an end point.

A binary sequence can be used to represent the trees; during training the sequence consists of a set of triplets - one for each node. If the third digit in a triplet is a 1, a category 0 path has passed through the node during the second step of the training. If a nodes has no branches, the first two digits are both 0. If there is a branch to the left and none to the right, the first digit is a 1 and the second is a 0. If there is only one branch and it is to the right, the first digit is 0 and the second is a 1. Both digits are 1 if there are two branches (we call these branch digits).

A binary sequence is comma free; i.e., it is not necessary to mark the beginning of each node layer, and the number of nodes in each layer need not be recorded. For example, take the binary sequence for tree d

```
110   110  100   110  010  010
      ↑          ↑
   Layer      Layer
     2          3
```

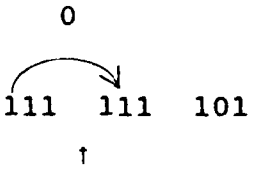
Both digits in the first triplet are 1 so there are two branches and two nodes with two triplets in the second layer. There are a total of three branch digit 1's in the second layer triplets, so there are

three nodes and triplets in the third layer. There are four branch digits in the third layer and thus four nodes in the bottom layer. There is no need to record bottom layer triplets because they are always (000).

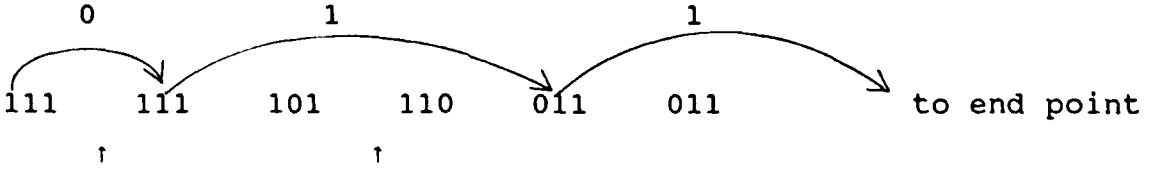
Tracing a binary path through this tree can be illustrated by tracing the path that is shown for tree h when the input is

$$\begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

The first three triplets are



The curve shows the path from the first node to the second; it starts on the 0 digit and ends on a 1 branch digit because X2 is 1. The next parts of the path are

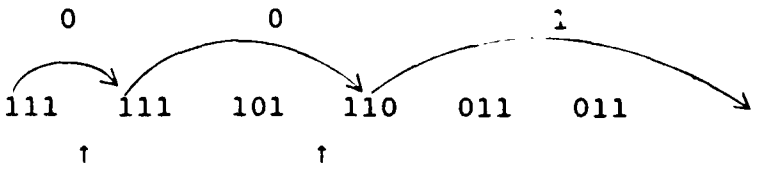


The second curve leaves from the 1 position of the first triplet in the second layer (node 2) and lands on the 1 position of the second triplet in the third layer because X2 and X3 equal 1. The third curve will go to an end point so the output of the tree is 1.

The tree pruning can be done conveniently using the binary sequence. All of the category 1 vectors are input and the first triplet on each

path that is not an end point, but has a third digit equal to 0, are changed to (001) and the path is terminated. To illustrate input

$$\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$



Node 4 is set to an end point; then

111 111 101 001 011 011

Where we have set the third digit of triplet 4 to 1 to indicate that the node designation has been changed to an end point. Finally, all triplets with the third digit equal to 0 are removed from the binary sequence. The third digits of the remaining triplets also can be removed (as in tree i) since they are no longer needed.

If the trees are implemented in software, tight logic can be written to generate, search, and prune the binary sequence. Simple digital binary sequence controlling logic can be constructed if arrays of trees are implemented on a special purpose parallel processor.

This training method requires two passes through the category 1 training set and one pass through the category 0 set. If an array of trees is to be built in parallel, then there would be three passes through the complete training set since the category 1 and category 0 sets would be different for each tree.

If there is insufficient memory to store the tree at the end of step one, the tree can be constructed in layers. For example, the category 1 paths could be run down to the fifth layer of nodes, and

then there could be an initial pruning. The paths would then be extended to layer ten, etc.

If the training set is "noisy", the number of nodes can be reduced as follows. The number of times a category 0 path and the number of times a category 1 path passes through each node is recorded. Then a node is checked in the pruning process only if the ratio of category 0 to category 1 paths at that node is greater than a threshold. If a node is checked but percentage of paths that branched to that node from the previous node is below a threshold, the check is removed. Then the pruning is done as before. This is an effective procedure for removing outlyers.

There is one other very important step in the tree construction; it is done before the actual construction to determine the order in which the elements of the input vector are entered into the tree. Enormous reduction in tree size when the vectors have large dimension and when the vector sets are large are possible by optimizing the order. Ordering should always be used for moderate and larger sized problems.

There are several methods to determine the order. They all make use of the fact that some input vector elements have much more statistical variability than others when there is clustering. The least variable elements are entered first. For example, some of the elements of each binary vector in a single cluster will be constant. Consider the cluster

$$\begin{pmatrix}0\\1\\1\end{pmatrix}\ \begin{pmatrix}1\\1\\1\end{pmatrix}\ \begin{pmatrix}1\\1\\0\end{pmatrix}\ \begin{pmatrix}0\\1\\1\end{pmatrix}$$

where the second element is constant (this cluster occupies the back plane of a three dimensional cube). The elements that are most often constant over the training set are entered first into the tree.

Error Correction

In Section I we noted that it was sometimes possible to determine when a classification error had occurred and to perform an alternate test that would improve the classification error rate. One method of determining the order makes this particularly easy to do.

A binary number can be converted back to decimal by multiplying the binary digit by 2^{i-1} where the digit is the ith least significant bit (LSB). For example

$$1011 \rightarrow 1\times2^3+0\times2^2+1\times2^1+1\times2^0=11$$

If some noise is added to the decimal number before the binary conversion and the first bit that is chanced is the ith LSB we can see that the magnitude of the noise was less than or equal to 2^i-1. For example, if the third bit in the binary number above was changed to 0 (i=2) we would know that the magnitude of the noise was less than 3.

To illustrate the next point assume, that a tree has been trained to recognize pattern vectors that have 10 decimal elements. Each element is converted to a binary word of say eight bits so that the binary pattern vector is 80 bits long. Reorder the bits so that the first 10 are the most significant bit (MSB) from each of the binary conversions of the 10 decimal values; the second 10 bits are the 10 second MSBs; and the last 10 bits are the 10 LSBs, etc.

If the tree has been trained with a category 1 vector, x_j, there is a path in the tree that leads to an end point that is followed by x_j. Suppose that there was noise added to the decimal version of x_j to form binary vector x'_j that was different from x_j. When x'_j was input to the tree, suppose it followed the x_j path to the 63rd level where it exited the path. This would be at a

level where the significance of the bit is 2^3. Thus none of the decimal elements of x'_j differ from the corresponding elements of x_j by more than 2^3-1, and

$$|x_j - x'_j| < \sqrt{10(2^3-1)}$$

Suppose the "not-signal" tree (as described in Section II) path that was followed by x'_j was exited at level 23 were the significance was 2^5. If this path was placed into the tree by non-signal vector N_R, then

$$|N_R - x'_j| < \sqrt{10\ (2^5-1)}$$

N_R is the non-signal training vector that is closest to x'_j.

If we know nothing about the binary elements of x_j below level 63 additive noise could have caused any distance between x_j and x'_j in the range of 0 to $7\sqrt{10}$ with equal probability. Similarly the distance between x'_j and N_R is uniformally distributed over the range 1 to $31x\sqrt{10}$. Therefore the best guess when both the signal and non-signal trees outputs are 0 is to choose the signal classification because the probability is approximately .75 that x'_j that is nearest to x_j.

It can be shown that as the size of the training set increases the accuracy of the tree-pair approaches the accuracy of the nearest neighbor rule which is well known to approach the accuracy of the Bayes decision rule. However, the computation and memory requirements of the trees can be vastly less which is the main argument for using neural networks instead of the nearest neighbor rule for pattern recognition in the first place.

APPENDIX B

This appendix shows examples of tree and binary sequence building.

The category 1 vectors are:

0 0 0 0 0 1 1
0 0 0 0 1 0 0
0 0 1 1 1 1 1
0 1 0 1 0 0 1

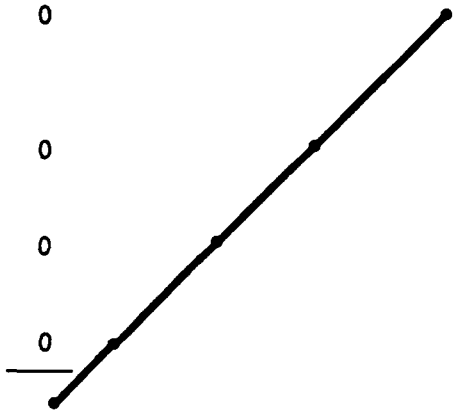
The category 0 vectors are:

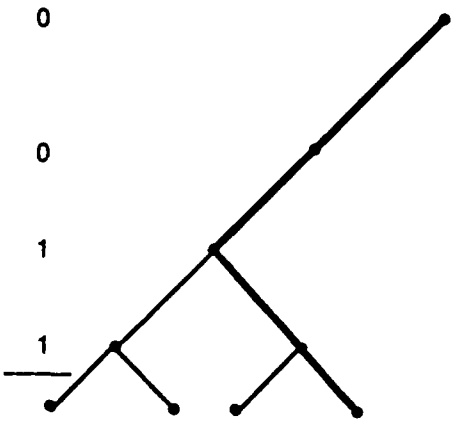
0 0 1 1 1 1
1 1 1 0 0 1
0 0 0 0 0 1
0 1 0 1 0 1

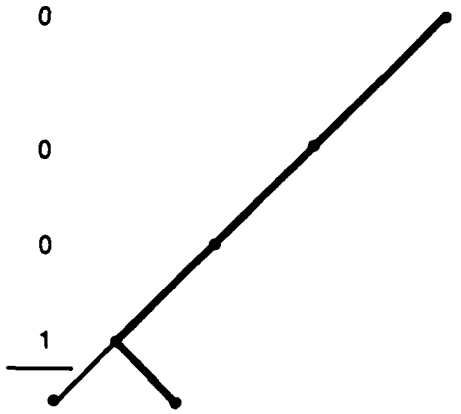
(1) <u>The Tree</u>

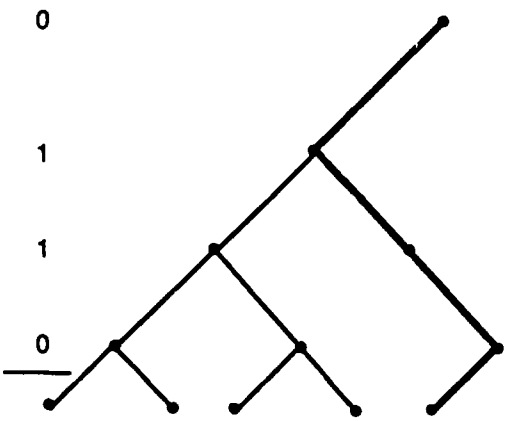
First cycle sequentially through the category 1 set as follows: (The bold line is the path followed with the current vector)

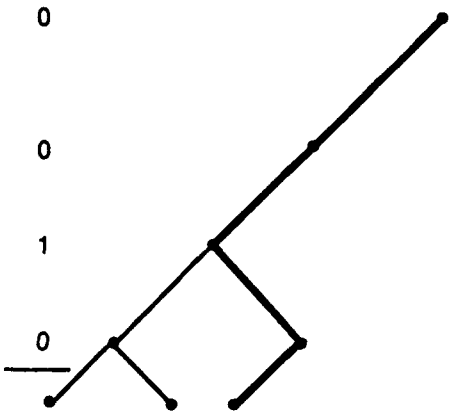
INPUT VECTOR:

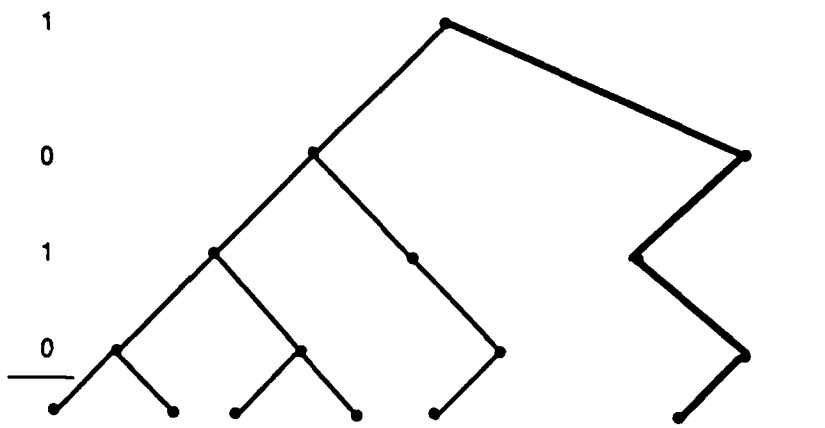


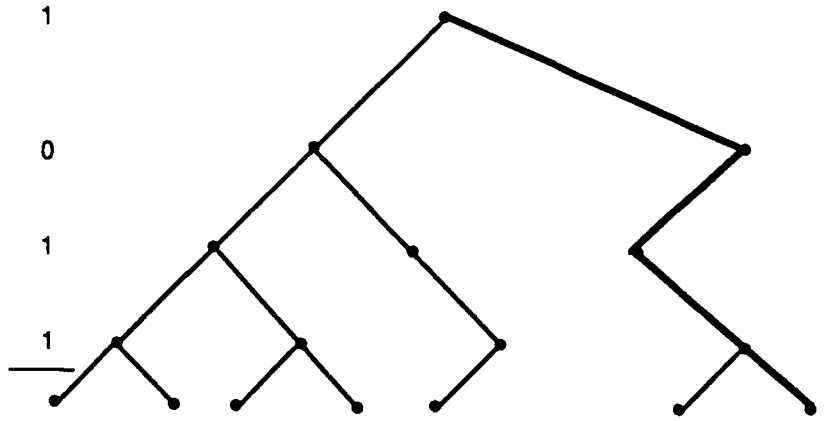




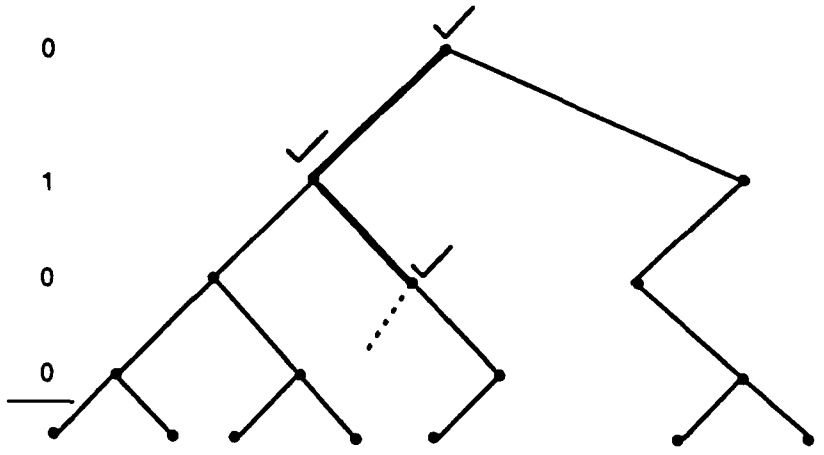


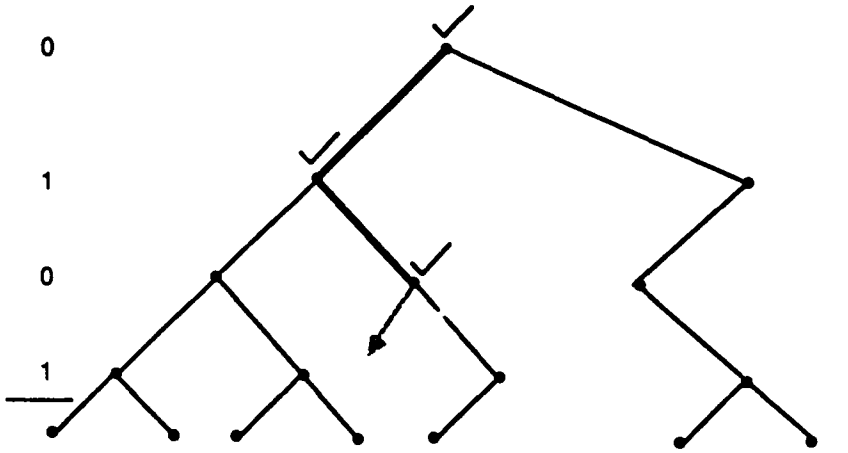


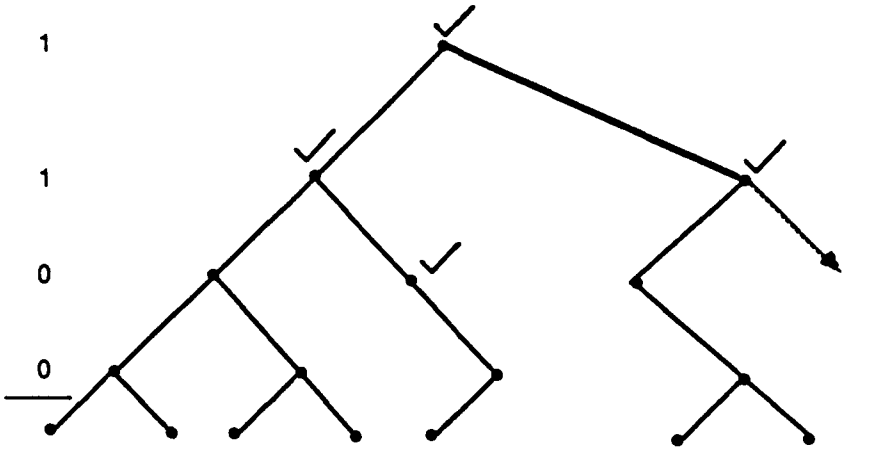


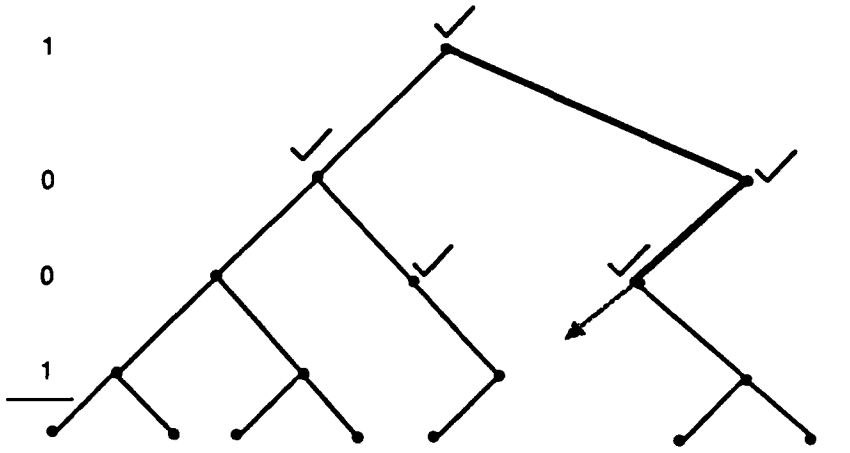


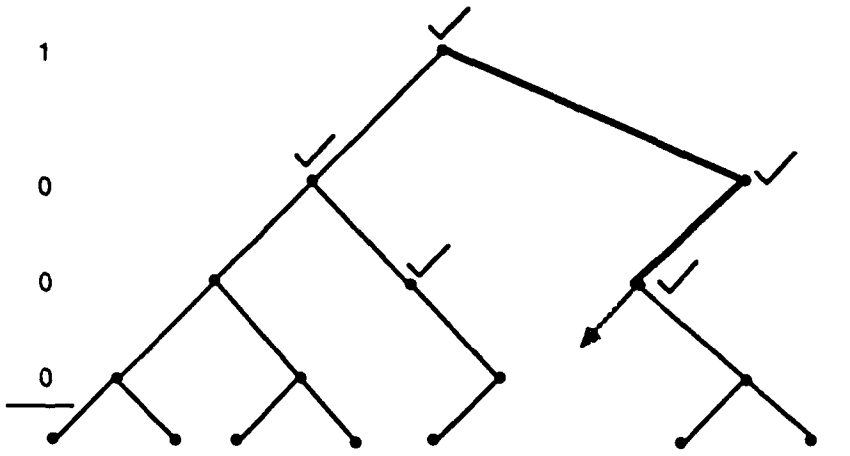
Now activate the nodes (an activated node is denoted by a check mark).

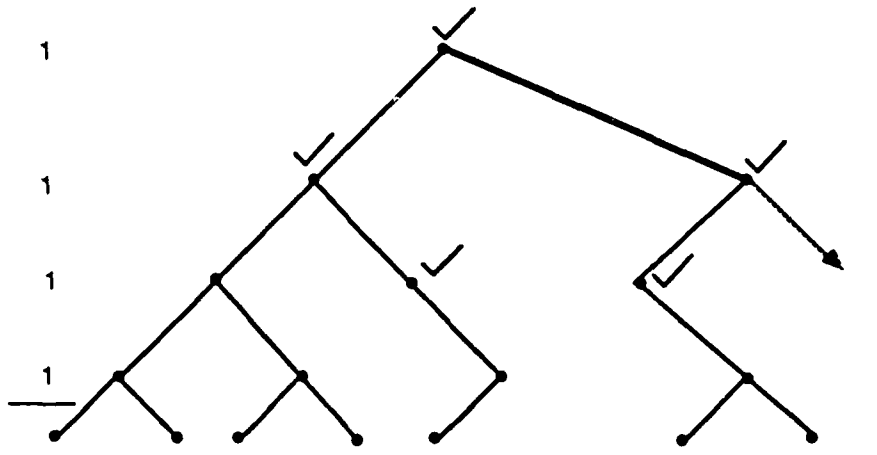



0
1
0
1


1
1
0
0


1
0
0
1

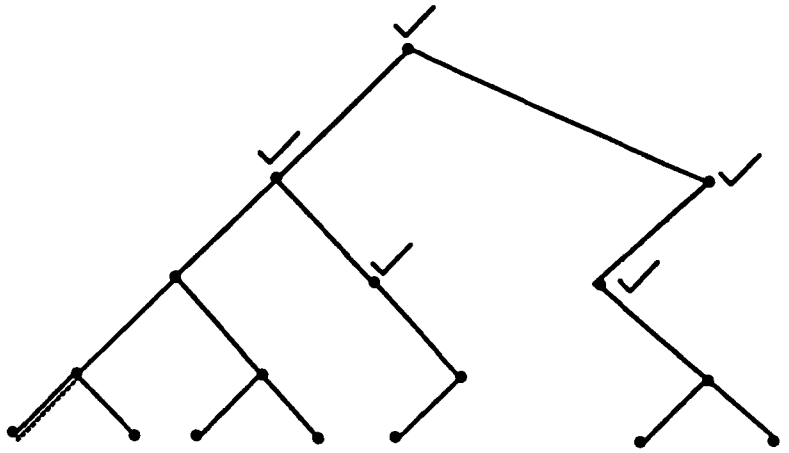


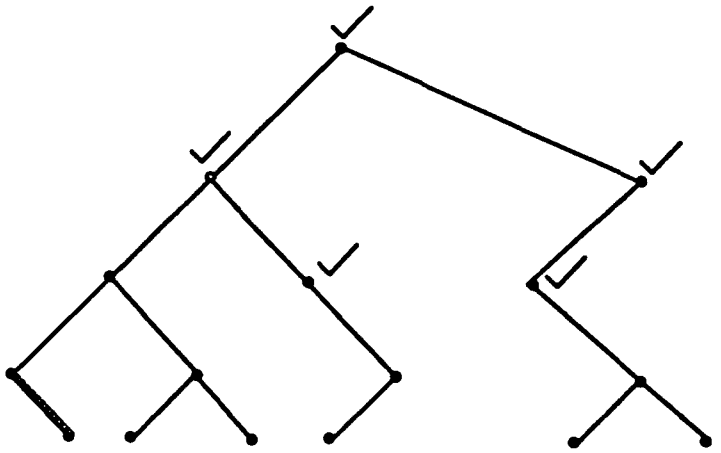


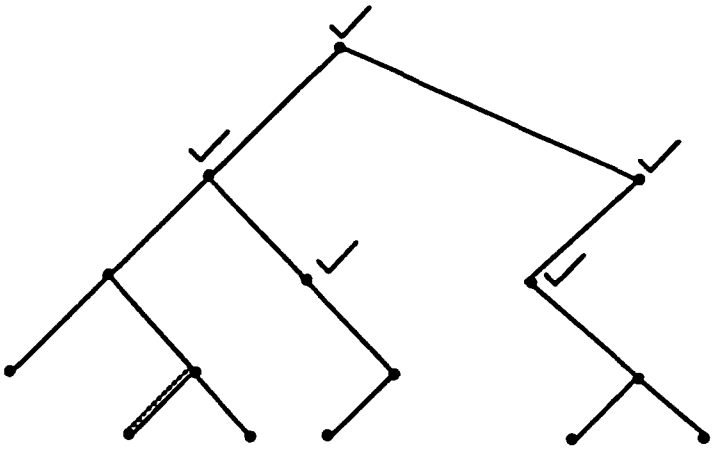
The next step is the pruning.

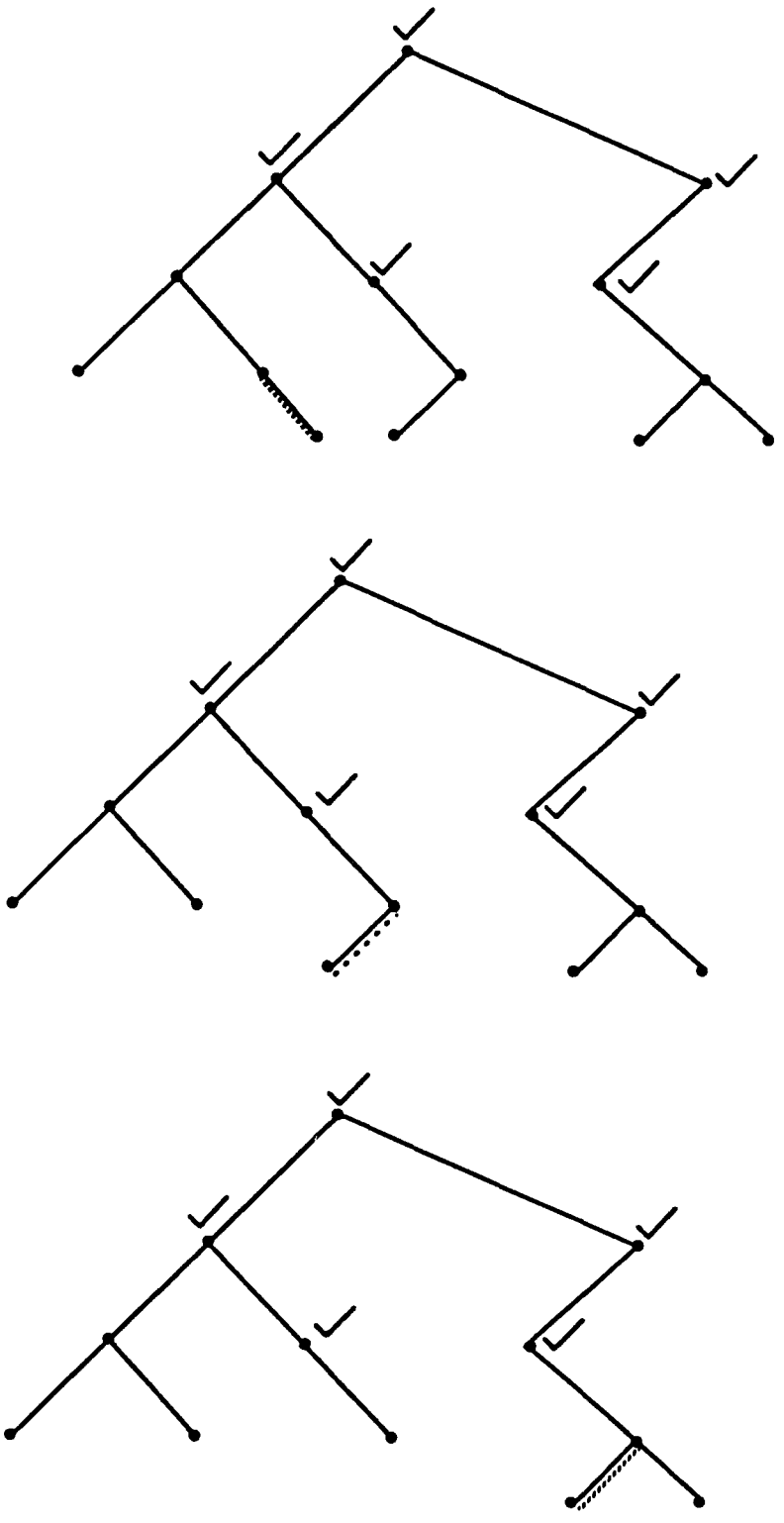
Rule: Start with node at the end of a path. If the parent node is not activated, then remove the daughter node. Start with all the nodes in the bottom row and work up row by row.

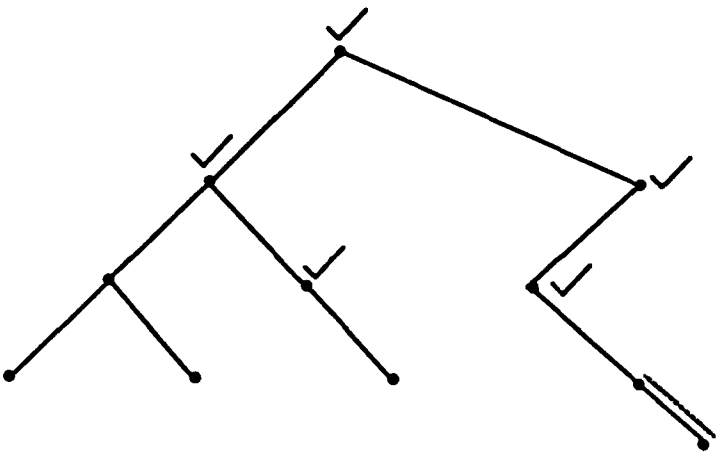
(The dashed line denotes the branch that is removed).

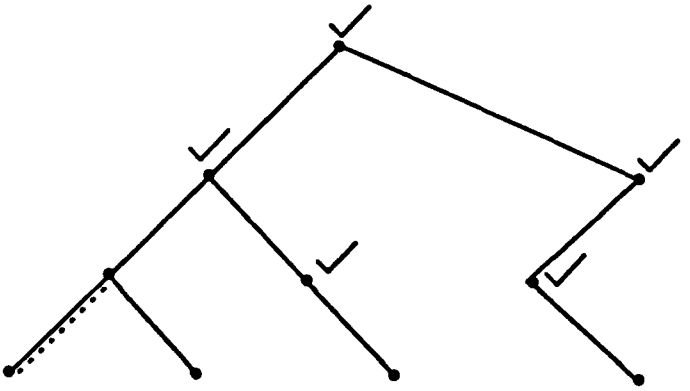




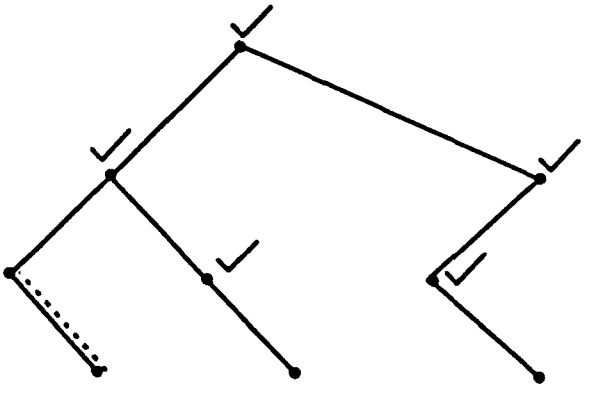


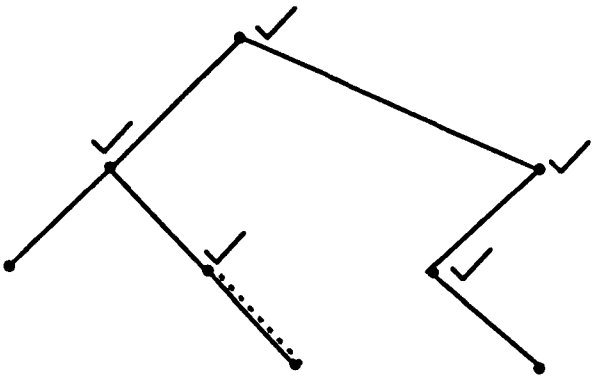




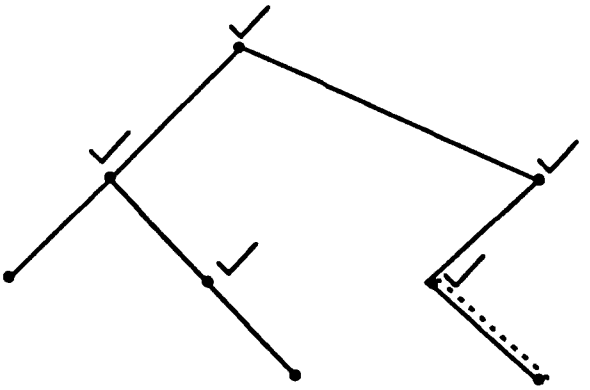


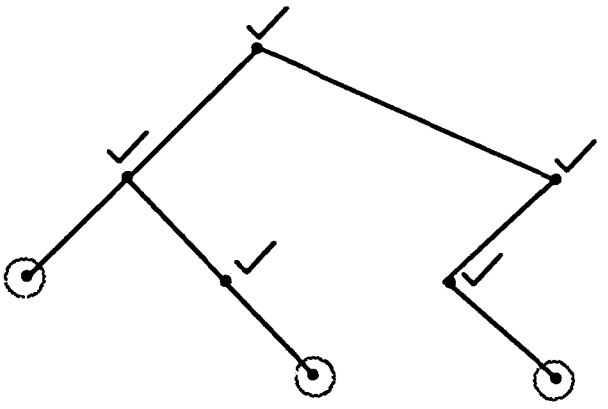
Next Prune from the next row.





The Dashed branch is not removed because it is connected to an activated node.

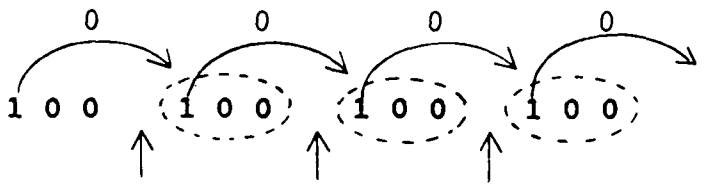




The end of each path is designed as an end point.

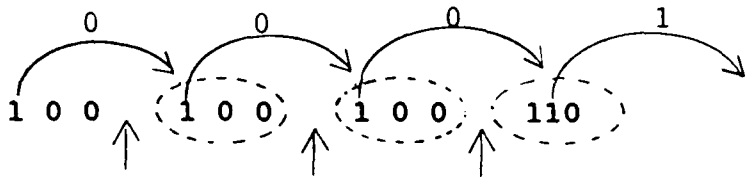
The second example will be the construction of the corresponding binary sequence.

Input Vectors: 0 0 0 0



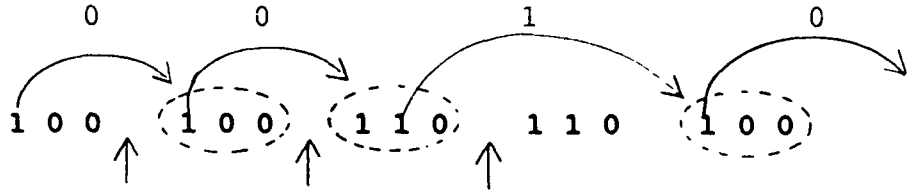
Row Positions

0 0 0 1



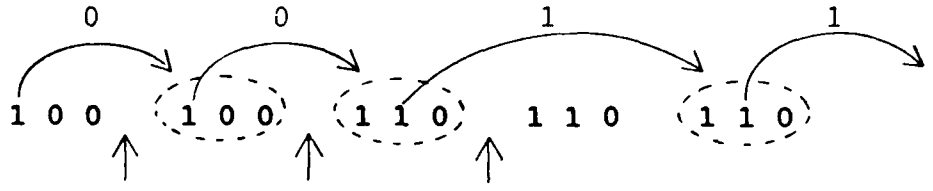
Change the last triplet because the departure is via a 1.

0 0 1 0

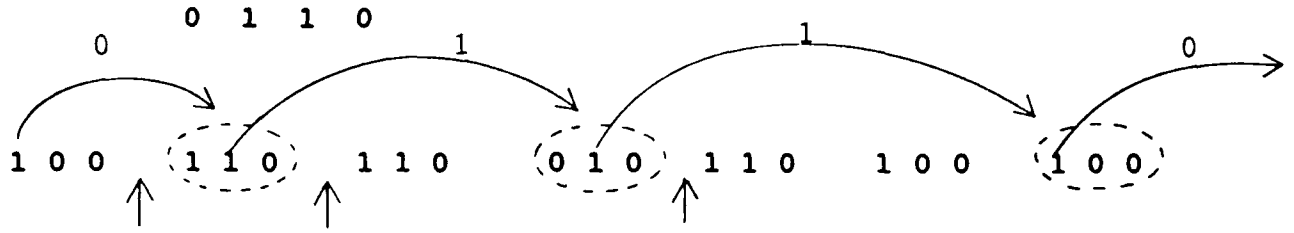


The third triplet changed to 110 because the departure is via a 1 and the second triplet is added to the last row because the third departure is from the second 1 in row 3.

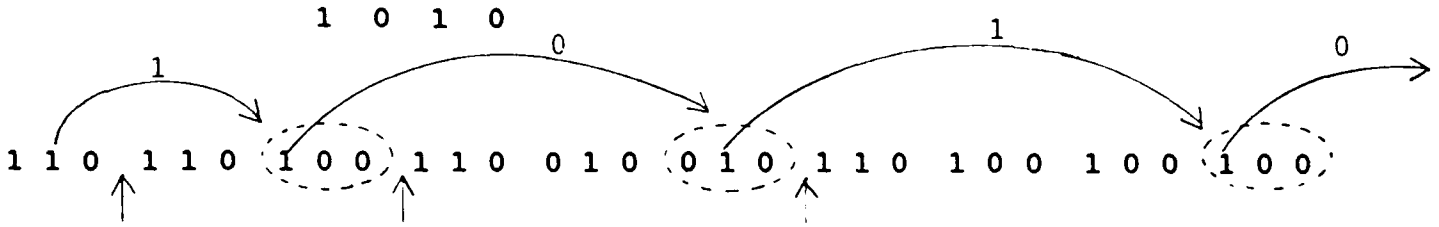
0 0 1 1



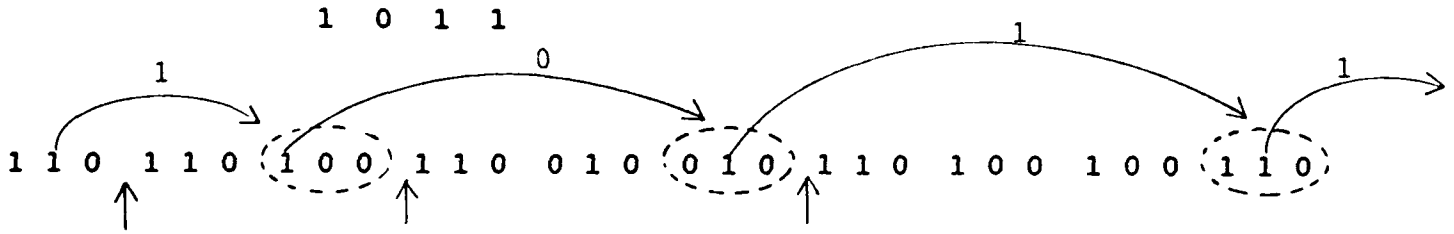
The second triplet in the last row is changed to 110 because the departure is via a 1.



The second triplet is changed to 110 because departure is via a 1 which is the second 1 in the second row. Therefore there must be another triplet in the third row. It is 010 since departure is via a 1 which is the third one in that row. therefore a third triplet must be added to the last row.



The departure is via 1 from the first triplet which is changed to 110 so that a second triplet, 100, must be added to the second row. The departure from this row is via a 0 and is from the third triplet, 010, must be added tot he third row. Departure from the third row is from the fourth 1 so that a fourth triplet, 100, must be added to the last row.



The last triplet is changed to a 110 because departure is via a 1.

This is the complete binary sequence.

Summary of Binary Sequence Building Rules:

The category 1 training vectors are presented once in sequence.

1. Whenever a triplet is changed from a 100 or a 010 to a 110, or a triplet is added to a row, a triplet must be added to every following row.

2. The new triplet will be a 100 if the departure from the new triplet row is via a zero and 010 if it is via a 1.

3. If there are n 1's to the left of the departure 1 in the departure row, then the new triplet is inserted to the right of the nth triplet in the next row.

4. If the vectors have N dimensions, there are N rows existing.

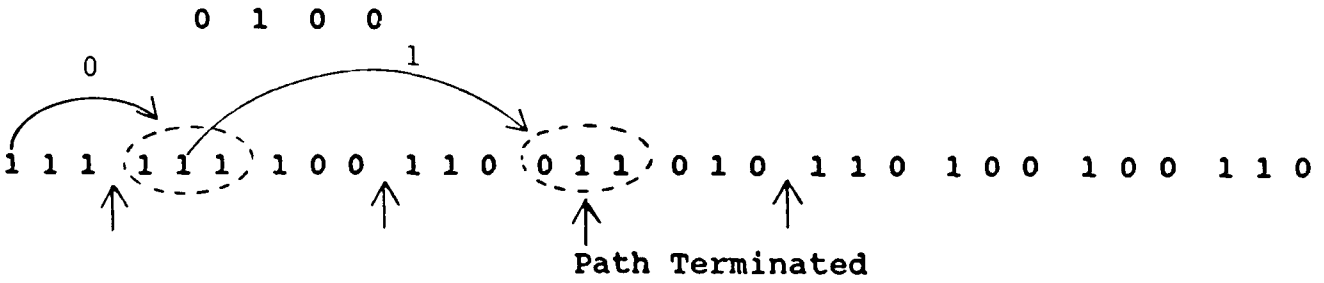
5. An existing triplet is changed from 010 to 110 when departure is via a 0 and from 100 to 110 when departure is via a 1.

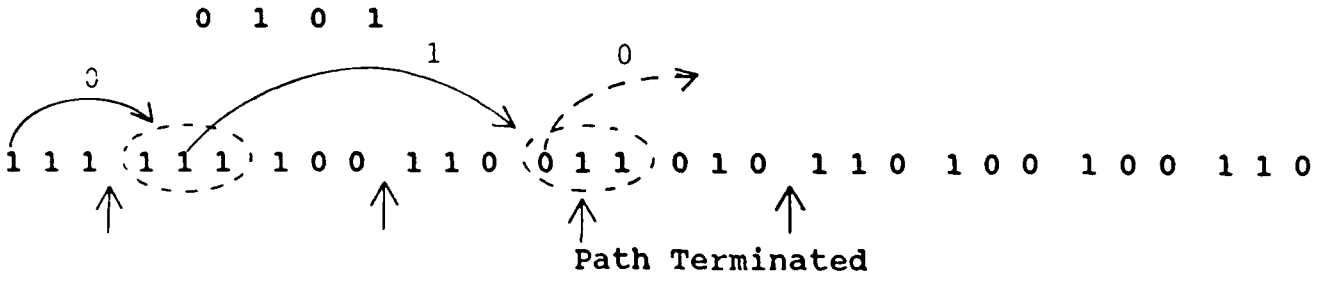
Triplet Activation Rules:

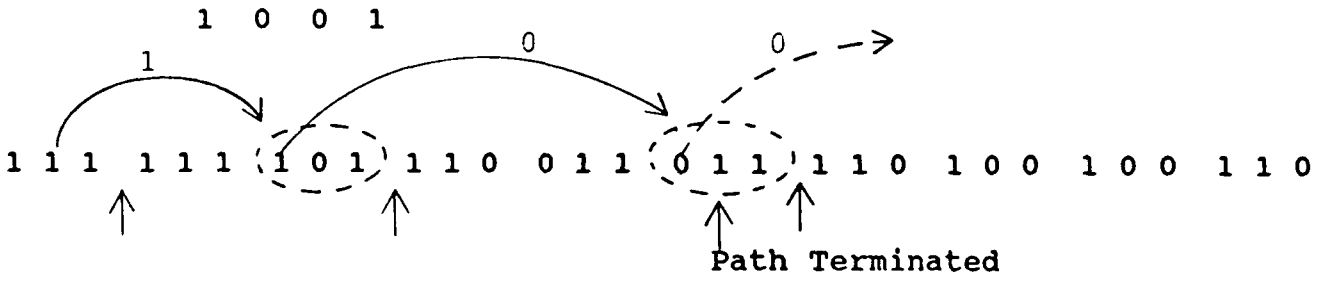
1. The category 0 vectors are presented once in sequence.

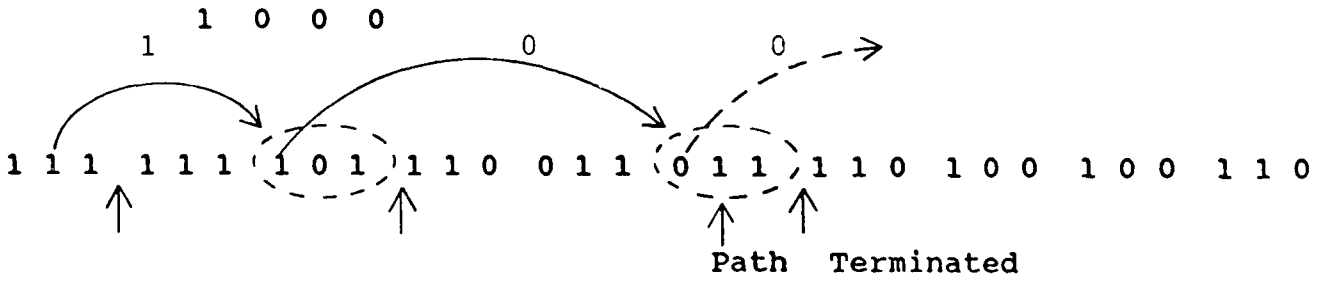
2. If a path reaches a triplet the third digit of the triplet is changed from a 0 to a 1 which signifies that the triplet is activated.

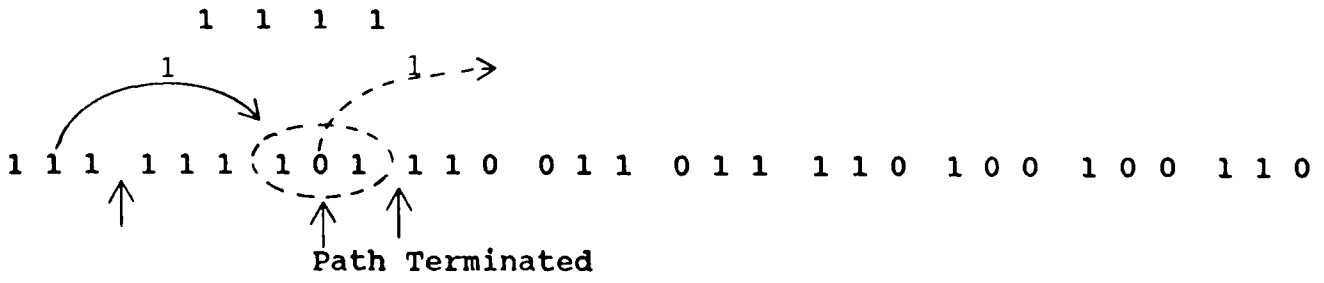
3. If a path reaches a 010 triplet and departure is via a 0 the path is terminated. If a path reaches a 100 triplet and departure is via a 1 the path is terminated. In both cases the third digit is changed to a 1.

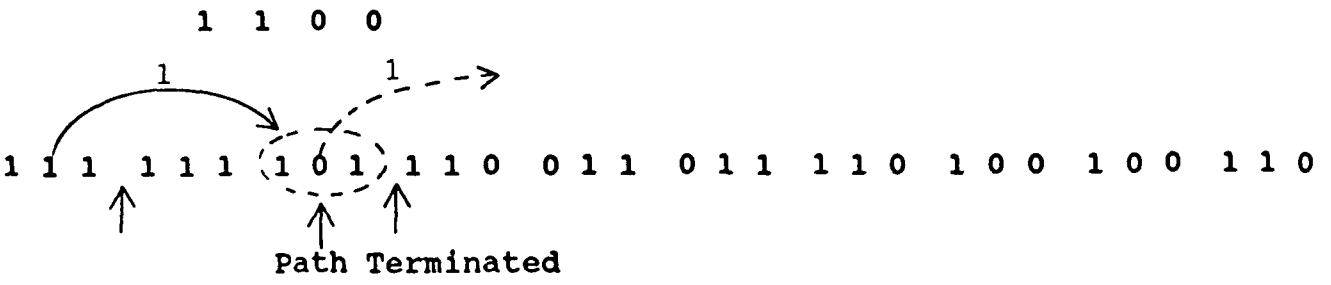

0 1 0 0
0
1
1 1 1 1 1 1 1 0 0 1 1 0 0 1 1 0 1 0 1 1 0 1 0 0 1 0 0 1 1 0
Path Terminated


0 1 0 1
0
1
0
1 1 1 1 1 1 1 0 0 1 1 0 0 1 1 0 1 0 1 1 0 1 0 0 1 0 0 1 1 0
Path Terminated


1 0 0 1
1
0
0
1 1 1 1 1 1 1 0 1 1 1 0 0 1 1 0 1 1 1 1 0 1 0 0 1 0 0 1 1 0
Path Terminated


1 0 0 0
1
0
0
1 1 1 1 1 1 1 0 1 1 1 0 0 1 1 0 1 1 1 1 0 1 0 0 1 0 0 1 1 0
Path Terminated


1 1 1 1
1
1
1 1 1 1 1 1 1 0 1 1 1 0 0 1 1 0 1 1 1 1 0 1 0 0 1 0 0 1 1 0
Path Terminated



Note: There are five activated triplets and in the tree there were five activated nodes.

Pruning Rules:

1. The nth daughter triplet in a row is connected to the parent triplet that contains the nth branch 1 in the row that is directly above.

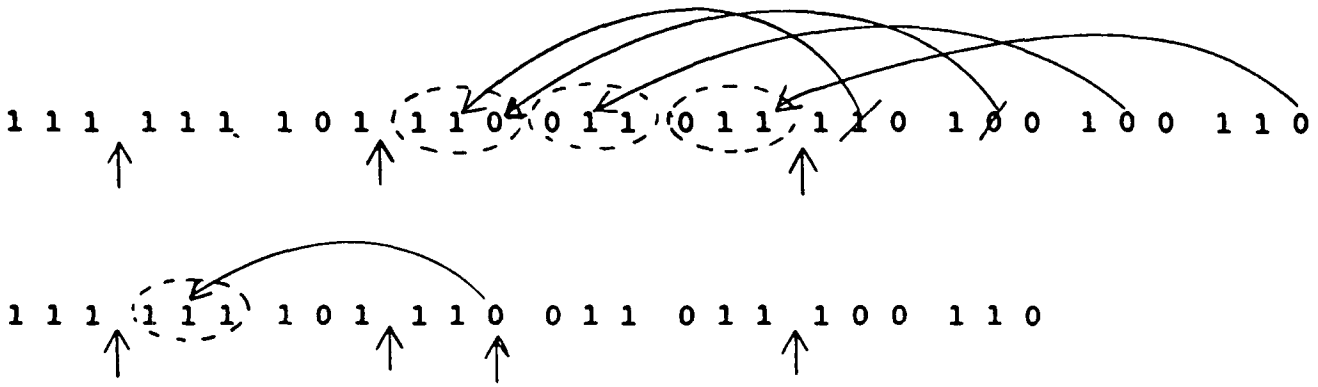
2. In a given row test all daughter triplets that are not activated. If such a daughter triplet is not also connected to an activated parent triplet remove it from the row. Start with the bottom row and work upwards.

3. After pruning, all remaining unactivated triplets will be set to 000.

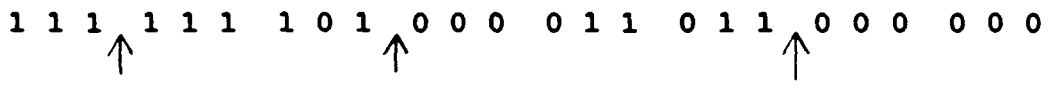
4. Drop the third digit in each triplet. The end points will now all be 00.

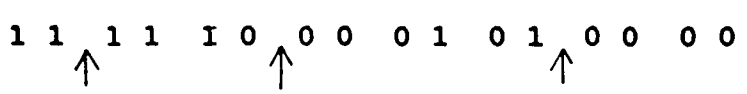
If a path reaches an end point the network output will be 1. If a path reaches 01 and departs via a 0, or reaches a 10 and departs via a 1, the output will be 0. Terminate the path.

If a path reaches a 01 in the bottom row and departs via a 1, or reaches a 10 in the bottom row and departs via a 0, the network output will be 1.



Not removed because
its parent is activated.





This is the final (and minimum length) binary sequence.

This is the binary representation of the final tree.

It takes one pass of the category 1 and one pass of the category 0.

Phase I Final Report Furnished To:

Dr. Thomas E. J4Hanna
Naval Medical Research Laboratory
Submarine Systems Department
Naval Submarine Base-New London
Box 900
Groton, CT 06349-5900

Dr. Mien Mann
Code 7033
Naval Air Development Center
Warminster, PA 18974-5000

Mr. Paul S. Rau
Code U-33
Naval Surface Weapons Center
White Oak Laboratory
Silver Springs, MD 20903-5000

Dr. Michael Rousseau
Code 2153
Naval Underwater system Center
New London Laboratory
New London, CT 06320

Dr. Steven Spiedel
Code 632
Naval Ocean systems Center
San Diego, CA. 92152

Dr. Harold Hawkins
Code 1142PS Poom 823
Office of Naval Research
800 N. Quincy Street
Arlington, VA. 22217-5000

Office of Naval Research
ONR SBIR Administrative Office
Code 11SP Room 811
800 N. Quincy Street
Arlington, VA 22217-5000

Dr. Barbara Yoon
DARPA
Information Science & Technology Office
1400 Wilson Blvd.
Arlington, VA 22209

Office of Naval Research
Code 1142BI
Attn: T. McKenna
800 N. Quincy Street
Arlington, VA 22217-5000

Administrative contracting Officer
Attn: SCD-C Code SO507A
DCASMA San Francisco
1250 Bayhill Drive
San Bruno, CA 94066-3070

LCdr. Casteen, USN
Naval Education & Training Support
Center-Paci
NAVTAD
San Diego, CA 92147-5070

Defense Technical Information Center
Bldg. 5
Cameron Station
Alexandra, VA 22314

Director, Naval Research Laboratory
Attn: Code 2627
Washington, DC 20375-5000

Office of Naval Research
Code 1114SE
Attn: C,G. Lau
800 N. Quincy Street
Arlington, VA 22217-5000